UNIVERSIDADE FEDERAL DO RIO DE JANEIRO

INSTITUTO DE MATEMÁTICA

CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

DANIEL ARTINE MASTUB

WILLIAM SATHLER LACERDA

SISTEMA DE INTEGRAÇÃO DE DADOS: UM ESTUDO DE CASO SOBRE VINHOS

RIO DE JANEIRO

2021

DANIEL ARTINE MASTUB
WILLIAM SATHLER LACERDA

SISTEMA DE INTEGRAÇÃO DE DADOS: UM ESTUDO DE CASO SOBRE VINHOS

Trabalho de conclusão de curso de graduação apresentado ao Departamento de Ciência da Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Orientadora: Profa. Giseli Rabello Lopes, D.Sc.

RIO DE JANEIRO
2021

Mastub, Daniel Artine
M423s Sistema de integração de dados: um estudo de caso sobre vinhos / Daniel Artine Mastub, William Sathler Lacerda. – 2021.

108 f.

Orientadora: Giseli Rabello Lopes.

Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) - Universidade Federal do Rio de Janeiro, Instituto de Matemática, Bacharel em Ciência da Computação, 2021.

1. Recuperação de informação. 2. Pareamento de registros. 3. Coleta de dados na web. 4. Sistema de recomendação. I. Lacerda, William Sathler. II. Lopes, Giseli Rabello (Orient.). III. Universidade Federal do Rio de Janeiro, Instituto de Matemática. IV. Título.

DANIEL ARTINE MASTUB
WILLIAM SATHLER LACERDA

SISTEMA DE INTEGRAÇÃO DE DADOS: UM ESTUDO DE CASO SOBRE VINHOS

Trabalho de conclusão de curso de graduação apresentado ao Departamento de Ciência da Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Aprovado em 08 de junho de 2021 .

BANCA EXAMINADORA:

Giseli Rabello Lopes

Profa. Giseli Rabello Lopes, D.Sc. (UFRJ)

Participação por videoconferência

Profa. Valeria Menezes Bastos, D.Sc. (UFRJ)

Participação por videoconferência

Profa. Maria Luiza Machado Campos, Ph.D. (UFRJ)

AGRADECIMENTOS

Um agradecimento especial à professora e orientadora Giseli Lopes pelos conhecimentos passados e suporte dado durante a orientação deste projeto. E também a todos os nossos colegas da UFRJ que estiveram com a gente nessa jornada, em especial aos amigos que ingressaram em 2015.1 e os da DevMob.

Daniel Artine Mastub

Agradeço primeiramente à minha mãe Amélia e ao meu pai Evaristo por todos os seus esforços para que este momento fosse possível. Ao longo deste caminho, também tive suporte e encorajamento de todos meus amigos e amigas de longa data. Por fim, este momento também não seria possível sem a contribuição e participação de meus amigos Leonardo, Mateus, Silvio, Vitor e William. Obrigado a cada um de vocês.

William Sathler Lacerda

Gostaria de agradecer aos meus pais Glaucio e Soraya, e minha irmã Camilla, que sempre me apoiaram em todos os momentos da minha jornada acadêmica e profissional; aos meus amigos da UFRJ Daniel Artine, Vitor Trentin, Silvio Mattos, Leonardo Dagnino e Matheus Villas Boas, que estiveram comigo ao longo dessa jornada e me acompanharam em inúmeras horas de estudo e trabalho árduo; aos meus amigos Eduardo Brito e Yan Matheus que sempre estiveram presentes para me ajudar com assuntos da faculdade ou pessoais; a minha equipe da Stone Age, em especial meu líder Mauro Miers, que me acolheu e com quem aprendi muito; e finalmente a minha namorada Beatriz Spaltemberg que esteve ao meu lado desde o início, celebrando todas as conquistas e ajudando nos momentos difíceis.

RESUMO

Com a quantidade crescente de dados sendo disponibilizados, especialmente na Web, é comum observar a representação de mesmas entidades em diferentes fontes de dados. Neste contexto, o objetivo deste trabalho consiste em desenvolver um sistema de integração entre os vinhos de diversas revendedoras com os do site Vivino. Assim, visa-se auxiliar um usuário que deseje pesquisar vinhos no Vivino, e comprá-los em uma revendedora local, sem precisar manualmente ir em cada uma verificar a disponibilidade do mesmo. Para alcançar tal objetivo, foram explorados temas como coleta de dados, pareamento de registros e sistemas de recomendação. Na aplicação desenvolvida, é possível para o usuário consultar todos os vinhos catalogados, selecionar algum individualmente e visualizar características como tipo de uva, vinícola, avaliação média e *link* de redirecionamento para as diferentes plataformas onde informações sobre o mesmo estão disponíveis. Além disso, diferentes critérios de filtragem também podem ser aplicados para atender uma necessidade mais específica e outros vinhos relacionados são recomendados.

Palavras-chave: Recuperação de Informação. Pareamento de Registros. Coleta de Dados na Web. Sistemas de Recomendação.

ABSTRACT

With the increasing amount of data being made available, especially on the Web, it is common to observe the representation of the same entities in different data sources. In this context, the objective of this work is to develop an integration system between the wines of several resellers with those of the website Vivino. Aiming to assist a user who wants to search for wines in Vivino, and buy them at a local dealer, without having to manually go to each one to check the availability of the same. To achieve this goal, topics such as Web scraping, record linkage and recommender systems were explored. In the developed application, it is possible for the user to visualize all cataloged wines, select one individually and view characteristics such as grape type, winery, average rating and links to the different platforms where information about it is available. In addition, different filtering criteria can also be applied to meet a more specific need and other related wines are recommended.

Keywords: Information Retrieval. Record Linkage. Web Scraping. Recommender Systems.

LISTA DE FIGURAS

LISTA DE CÓDIGOS

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
ER	Entidade-Relacionamento
DAO	Data Access Object
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
JS	JavaScript
JSON	JavaScript Object Notation
MVC	Model-view-controller
MVP	Mínimo Produto Viável
REST	Representational State Transfer
SGBD	Sistemade Gerenciamento de Banco de Dados
SQL	Structured Query Language
URL	Uniform Resource Locator
XML	eXtensible Markup Language

SUMÁRIO

1 INTRODUÇÃO

Atualmente, por conta do crescimento da quantidade de dados sendo disponibilizados, é comum observar diferentes fontes de dados representarem uma mesma entidade. Tais representações podem divergir nas informações disponibilizadas, sua forma de estruturação e seus conteúdos, o que torna o processo de comparação e identificação dessas diferentes representações de uma mesma entidade ainda mais complexo.

Os diferentes desafios que surgem neste contexto estão associados à integração de dados, sendo que as soluções propostas tradicionalmente abordam as seguintes etapas principais [9]: (i) *alinhamento de esquemas*, que endereça os desafios de ambiguidade semântica e objetiva entender quais atributos têm os mesmos significados e quais não tem; (ii) *interligação de registros*, que endereça o desafio de ambiguidade na representação de instâncias, e objetiva entender quais registros representam as mesmas entidades e quais não; e (iii) *fusão de dados*, que endereça o desafio de qualidade de dados, e objetiva entender qual valor utilizar nos dados integrados quando as fontes possuem valores conflitantes.

Um cenário para visualização do problema pode ser exemplificado a partir da existência de um grande número de vinícolas e vinhos em todo mundo, onde cada vinho possui suas peculiaridades. Diversas plataformas possuem informações em suas bases, como por exemplo o Vivino[1]. O Vivino é um dos sites mais populares para consultar informações relevantes sobre vinhos. Neste, qualquer pessoa pode criar uma conta na plataforma e avaliar quaisquer produtos, o que contribui para a construção de uma base rica de avaliações. Isso permite que consumidores tenham subsídios para auxiliar em suas decisões de compras.

Por ter uma base de dados ultrapassando os 3 milhões de vinhos[2], o Vivino acaba não possuindo detalhes tão precisos e atualizados sobre os vinhos em toda sua base. Por conta disso, diversos consumidores recorrem aos sites de revendedoras

[1] https://vivino.com/

[2] https://vivino.com/wines

de vinhos. O motivo é que, por possuirem um inventário de vinhos menor, fica mais fácil construir uma base mais rica em quesitos técnicos sobre cada vinho. Porém, estas não possuem uma quantidade de usuários tão expressiva para ter uma boa base de avaliações, ou mesmo não possuem funcionalidades para avaliação dos vinhos. Consequentemente, vem à tona o problema citado no início dessa seção, onde é possível haver representações distintas para uma mesma entidade.

Neste cenário, um consumidor comum, que deseje analisar as avaliações do Vivino e cruzar isso com os detalhes que um vendedor provê, acaba se deparando com o árduo trabalho de ter que pesquisar pelo mesmo vinho em diversos sites. Precisando acessar cada site de uma revendedora individualmente, verificar se o mesmo possui o vinho que ele está procurando (e caso não possua, seguir repetindo essa etapa até encontrar o vinho em algum site), e, finalmente, após isso, ele pode analisar se as características do vinho são de seu agrado (isso ainda tentando estabelecer correspondências entre as informações do Vivino com as do site encontrado). Além disso, se ele quiser encontrar o melhor preço para o vinho, precisa buscar em todas as revendedoras para descobrir quais o possuem e qual o preço praticado.

Levando estes fatos previamente citados em consideração, no presente projeto, é proposta uma abordagem para integrar os dados sobre vinhos obtidos de diferentes revendedoras com os do Vivino. Isto visa facilitar a tarefa de um usuário que quer escolher um vinho, baseado nas avaliações e informações básicas do Vivino acrescidas da descrição normalmente mais rica fornecida pela(s) revendedora(s), incluindo onde possivelmente o vinho possa ser adquirido.

As revendedoras de vinho utilizadas para a realização deste trabalho foram a Winebrands[3], Grand Cru[4] e Evino[5], além do próprio Vivino. Estas fontes disponibilizam seus dados em diferentes formatos, como texto simples (*plain text*), JSON (*JavaScript Object Notation*) e XML (*eXtensible Markup Language*). Por conta disso, foi necessário desenvolver diferentes estratégias para coletar os dados a partir dessas fontes. Para coletar os dados necessário neste projeto, através da Web, foi

[3] https://winebrands.com.br/
[4] https://www.grandcru.com.br/
[5] https://www.evino.com.br/

realizado o estudo e implementação de diferentes estratégias para *Web Scraping* [18]. Para a integração dos dados entre as fontes, foi necessário comparar os vinhos do Vivino com os das revendedoras, buscando encontrar os pares de correspondência (pareamento). Para tanto, dentre as etapas principais das estratégias de integração, houve uma etapa inicial de mapeamentos entre os esquemas das diferentes fontes, seguida de uma etapa de interligação com definição e implementação de estratégias de *Record Linkage* [5] específicas para este projeto. A etapa de fusão de dados não foi contemplada no escopo deste projeto.

Além disso, foi desenvolvida uma aplicação Web para facilitar o usuário na busca por vinhos. Dentre as funcionalidades disponibilizadas estão: (i) listagem paginada do catálogo de vinhos; (ii) página contendo informações básicas sobre um vinho específico e links externos para os sites do Vivino e revendedora(s) em que este encontra-se disponível; (iii) filtros para seleção de diferentes características sobre os vinhos a serem exibidos; e (iv) recomendação de vinhos, utilizando abordagem de sistemas de recomendação [13] baseada em conteúdo, visando ajudar o usuário a conhecer novos vinhos similares ao que está sendo visualizado.

Por fim, as diferentes estratégias de pareamento foram avaliadas manualmente por alguns usuários, possibilitando melhorias incrementais (ver Capítulo 4). A avaliação da estratégia de pareamento de vinhos em sua versão final obteve um índice de acerto de aproximadamente 73%.

O restante do texto está estruturado da seguinte forma:

- No Capítulo 2, são explicados alguns fundamentos essenciais para o presente projeto, como coleta na Web (*Web Scraping*), pareamento de registros (*Record Linkage*) e Sistemas de Recomendação. Também são apresentados alguns trabalhos relacionados aos temas abordados.
- No Capítulo 3, é descrito o projeto e a população da Base de Dados proposta, abordando cada passo realizado desde a coleta de dados nas diferentes fontes, projeto do banco de dados até a população do mesmo.
- No Capítulo 4, é apresentado como os pares de registros que referenciam mes-

mos vinhos, coletados das diferentes fontes, foram identificados, detalhando cada etapa deste processo.

- No Capítulo 5, é descrito o *front end* que foi desenvolvido para possibilitar que os dados sejam exibidos para o usuário final, incluindo funcionalidades de filtros e recomendação de vinhos relacionados.

- No Capítulo 6, são apresentadas as conclusões sobre o desenvolvimento do projeto, trazendo dificuldades enfrentadas durante o processo, assim como possíveis ideias para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA E TRABALHOS RELACIONADOS

Este capítulo busca apresentar alguns conceitos e técnicas que foram fundamentais para o desenvolvimento desse projeto. As próximas seções discutem os seguintes temas: Coleta de Dados na Web (*Web Scraping*) (Seção 2.1), Pareamento de Registros (*Record Linkage*) (Seção 2.2) e Sistemas de Recomendação (*Recommender Systems*) (Seção 2.3). Além disso, também são apresentados alguns trabalhos relacionados (Seção 2.4), selecionados por abordarem pelo menos um desses temas.

2.1 COLETA DE DADOS NA WEB

Web scraping[1], ou coleta de dados da Web, é o processo de extração de dados de sites, por meio de processos automatizados, tipicamente para armazenamento visando posterior utilização ou análise. Em geral, essa técnica envolve analisar o código HTML (*HyperText Markup Language*) de páginas Web para extrair os seus dados ou, em certos casos, utilizar APIs (*Application Programming Interface*) internas que os sites usam para buscar suas próprias informações e montar suas páginas[2].

Segundo Mitchell [18], em uma definição mais restritiva do termo, *Web scraping* é definido como a prática de coletar dados por qualquer meio que não seja um programa interagindo com uma API (ou, obviamente, por meio de um ser humano usando um navegador Web). Dessa forma, é mais comumente realizado ao escrever um programa automatizado que envia requisições a um servidor Web, solicitando dados (geralmente na forma de páginas HTML e outros formatos de arquivos que compõem páginas Web) e, em seguida, analisa esses dados para extrair as infor-

[1] *Web Scraping* é um termo que pode ser utilizado para descrever até o processo de manualmente entrar em um site e copiar um texto contido nele. Porém, no contexto desse projeto, serão utilizadas as definições que associam o termo para descrever a ação de desenvolver um robô (*bot*) que será responsável por extrair certas informações de uma página Web.

[2] https://www.imperva.com/learn/application-security/web-scraping-attack/

mações necessárias. Essa maneira pode ser considerada um pouco mais trabalhosa, pois um arquivo HTML não é o mais apropriado para ser parseado a fim de extrair informações de uma maneira estruturada.

De maneira genérica, é possível definir o processo de coleta, análise e processamento através do pseudocódigo apresentando a seguir (Código 2.1).

Código 2.1: Pseudocódigo do algoritmo de coleta (adaptado de [2])

```
Coletando (paginas S){
FilaURL <-- S
faca {
        p <-- Selecionar-Proxima-URL(FilaURL)
        conteudo <-- Baixar(p)
        (texto, links, estrutura) <-- Extrair-E-Filtrar(conteudo)
        Armazenar(texto, links, estrutura)
        FilaURL <-- Remover-Elemento-Processado
 } enquanto (FilaPossuiElementos(FilaURL))
}
```

Um método de *Web scraping* como este, normalmente, envolve fornecer a um robô, uma fila inicial de URLs (*Uniform Resource Locator*) sementes que ele deverá visitar para realizar a coleta. Este, por sua vez, irá recuperar o código HTML de cada página e extrair os dados desejados baseado em um conjunto de regras previamente elaborado pelo seu desenvolvedor. Essas informações serão salvas, pondendo ser estruturadas, por exemplo, em um Banco de Dados Relacional ou uma planilha. Dentre as informações contidas em cada página visitada, pode-se extrair os *links* para outras páginas Web e adicioná-los à fila de URLs a serem visitadas.

Grandes sites normalmente possuem um mapa de suas URLs, conhecido como mapa do site (*sitemap*)[3], ou outros recursos para esses robôs. Referências para estes recursos normalmente estão listadas no arquivo `robots.txt`, disponibilizado no diretório raiz de tais sites. Este arquivo, além de indicar a localização do mapa do site, pode, muitas vezes, conter diversas outras informações (como restrições de acesso, tempo limite entre requisições, entre outros).

[3]Um mapa do site (ou *sitemap*) é uma lista de páginas dentro de um domínio.

Outro método de *Web Scraping* envolve analisar as requisições que um site faz ao disponibilizar certas páginas. Por exemplo, ao carregar uma página de um produto em um site de vendas, é feita uma requisição HTTP (*Hypertext Transfer Protocol*) para seu próprio servidor (APIs), a fim de buscar as informações do produto selecionado pelo usuário. Essas mesmas requisições podem ser utilizadas por um robô para buscar essas informações. Cada produto do site possui um identificador (`id`) único na base de dados que, muitas vezes, pode ser obtido através de um *scraping* em páginas contendo o catálogo dos produtos, ou até pelo mapa do site. Possuindo os `ids` de cada um dos produtos, é possível efetuar as mesmas requisições que o site faz para seu *back end* e buscar as informações de todos os produtos. Esse método pode ser considerado mais simples e confiável, visto que essas APIs retornam dados de forma estruturada, normalmente em JSON (*JavaScript Object Notation*) e XML (*eXtensible Markup Language*).

2.2 PAREAMENTO DE REGISTROS

De acordo com Fellegi et al. [11], o problema original a ser resolvido pelo pareamento de registros (*Record Linkage*) é comparar registros de diferentes bases de dados a fim de, por exemplo, fundi-los com o propósito de estender as informações acerca de determinado domínio. Dessa forma, a proposta consiste em, a partir de dados de diferentes fontes, descobrir aqueles que representam mesmas entidades. Estes dados a serem pareados podem ou não possuírem identificadores comuns nas diferentes fontes.

A fim de ilustrar o problema a ser revolvido com as técnicas de pareamento de registros, a Figura 1 apresenta um exemplo onde são representados registros de duas fontes distintas contendo informações sobre vinhos. Tanto na fonte A quanto na fonte B, os campos dos registros apresentam as mesmas informações, logo, bastaria apenas comparar campo a campo para garantir a igualdade entre os registros das diferentes fontes.

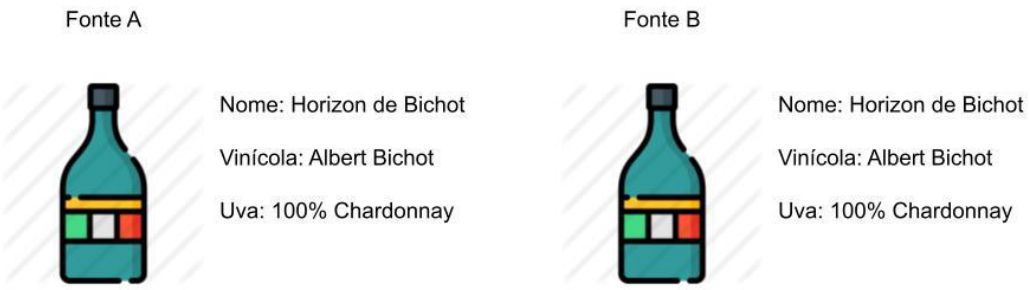

Figura 1: Vinhos de diferentes fontes com as mesmas informações.

Outra possível situação pode ser visualizada na Figura 2, onde mais uma vez o vinho da Fonte A representa o mesmo da Fonte B, porém com informações cadastradas nas bases de maneira diferente.

Figura 2: Vinhos de diferentes fontes e informações diferentes.

Diferentes técnicas podem ser aplicadas para o pareamento de registros, na tenta-

tiva de identificar registros de fontes distintas que representam uma mesma entidade. Dependendo do tipo de dado, seja ele texto, número ou data, diferentes algoritmos e técnicas podem ser utilizados para validar se determinados campos ajudam a identificar se um determinado registro corresponde ao mesmo registro de outra base. Neste cenário, além de comparar os campos por igualdade, funções de similaridade de *strings* podem ser aplicadas.

Além disso, as técnicas para pareamento de registros podem ser classificadas em dois grandes grupos [10]: determinísticas (ou deterministas) e probabilísticas. Nas seções seguintes, as técnicas determinísticas (Seção 2.2.1) e probabilísticas (Seção 2.2.2) serão discutidas.

2.2.1 Pareamento de Registros Determinístico

De acordo com Zhu et al. [23], a proposta da abordagem determinística é comparar um ou mais identificadores entre os registros provenientes de diferentes bases de dados. Caso todo o grupo de identificadores selecionado seja igual ao grupo de identificadores do registro de outra base, um pareamento foi encontrado. Com auxílio da Figura 1, é possível exemplificar este caso.

Outro exemplo pode ser ilustrado com auxílio da Figura 3, onde a Fonte B cadastrou a vinícola com um valor semelhante ao nome do vinho enquanto a Fonte A cadastrou o vinho com o nome original da vinícola. Neste caso, mesmo que os dois vinhos sejam do país “Argentina”, tenham como nome “Horizon de Bichot” e tenham informações sobre vinícolas diferentes, a abordagem determinística não acusaria ambos como par, o que pode não se fazer como verdade, visto que diferentes revendedoras podem cadastrar uma vinícola diferente, seja por critérios de digitação ou informações mal preenchidas. Apesar do erro na informação relativa à vinícola na Fonte B, o vinho cadastrado nas duas fontes é o mesmo (mesma entidade está sendo representada).

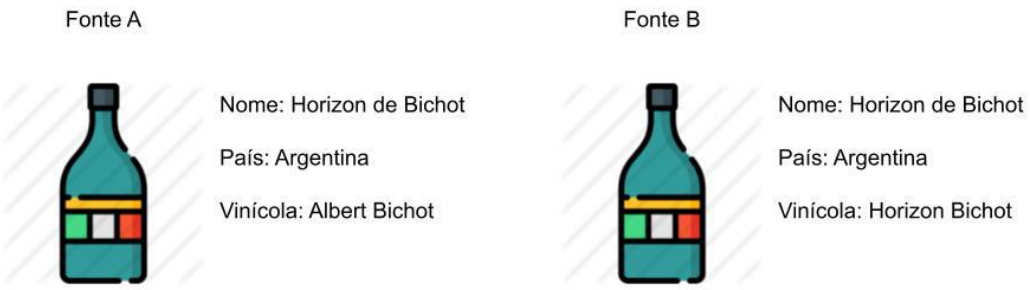

Figura 3: Vinhos de diferentes fontes e vinícolas diferentes.

Por conta desta limitação, faz-se necessário a utilização de um processo de *link* não-determinístico, onde é possível estimar a probabilidade de determinado par ser autêntico ou não.

2.2.2 Pareamento de Registros Probabilístico

Por conta da limitação de abordagens determinísticas, foram propostas abordagens probabilísticas para validar se determinados dados de diferentes fontes podem ser interligados ou não. Diferente da técnica determinística, o propósito não é ter como resultado se determinados dados representam ou não uma mesma entidade, e sim qual a probabilidade disto ser verdade.

A abordagem de Pareamento de Registros Probabilística tradicional, proposta por Fellegi & Sunter [11], utiliza probabilidades m e u. Sendo m a probabilidade de determinado par ser uma ligação autêntica, dados com maior confiança provêm uma probabilidade maior para m; e u a probabilidade de determinado par não ser uma ligação autêntica, onde tais probabilidades devem ser estimadas de maneiras diferentes conforme o domínio do problema a ser resolvido. Ainda de acordo com os

autores, uma das principais necessidades para o cálculo das probabilidades m e u é a possibilidade da introdução de erros e dados incompletos provenientes de erros de informação, digitação e transcrição, por exemplo.

De acordo com Fellegi & Sunter [11], é necessário que haja uma definição de pesos para cada um dos campos das diferentes bases. Por exemplo: se determinados dados acerca de vinhos contêm a mesma vinícola como registro, o peso desta informação não será tão relevante caso possua o mesmo campo de nome entre os registros. Um exemplo pode ser apresentado com o auxílio da Figura 4: neste, a igualdade do nome dos vinhos é três vezes mais significativa em relação ao nome do país. Ou seja, caso um par qualquer de registros proveniente de fontes diferentes concorde em relação ao nome do vinho e discorde no campo de país, a probabilidade de ser autêntico será maior se comparado a outro par qualquer que discorde em relação ao nome do vinho e concorde no campo referente ao país. Logo, em uma análise probabilística, o valor de m para o primeiro par será maior se comparado ao segundo.

Figura 4: Vinhos de diferentes fontes e vinícolas diferentes com pesos atribuídos.

Para o cálculo de probabilidades de pareamento entre identificadores, coeficientes como Sørensen–Dice, elaborado a partir da junção dos trabalhos de Sørensen [22] e Dice [8] podem ser utilizados para estimativas. O valor do coeficiente obtido é

utilizado para avaliar a similaridade entre dois conjuntos de dados quaisquer. Após a obtenção das similaridades, faz-se necessário estabelecer um valor de corte para considerar se determinado par de registros representa um pareamento autêntico.

Em sua proposta original, o coeficiente de Sørensen–Dice é calculado a partir da cardinalidade de dois conjuntos, podendo ser obtido por:

$$DSC(X,Y) = \frac{2|X \cap Y|}{|X| + |Y|} \tag{2.1}$$

onde X e Y representam conjuntos quaisquer. No âmbito de comparação de cadeias de caracteres, um exemplo pode ser dado utilizando nome de vinhos: dada duas cadeias de caracteres representando o nome dos vinhos “Pinot” e “Pinoti”, o cálculo de similaridade pode ser feito a partir da composição de bigramas. No caso do cálculo da equação 2.1 aplicado aos bigramas, |X| representa o número de bigramas únicos de X, |Y| representa o número de bigramas únicos de Y, e $|X \cap Y|$ representa o número de bigramas únicos compartilhados. Com a formação dos bigramas das duas cadeias de caracteres obtemos: o conjunto X, onde X = {“Pi”,“in”,“no”,“ot”} e o conjunto Y, onde Y = {“Pi”,“in”,“no”,“ot”,“ti”}. Aplicando o cálculo apresentado para obtenção do coeficiente, teremos o valor aproximado de 0,89.

A fim de obter a distância entre duas cadeias de caracteres, algoritmos como o Damerau-Levenshtein, elaborado a partir de trabalhos dos pesquisadores Frederick J. Damerau [6] e Vladimir Levenshtein [16], podem ser utilizados. A proposta deste é calcular o número mínimo de alterações (inserção, remoções, substituições ou troca de caracteres adjacentes) para que uma cadeia de caracteres seja considerada igual à outra. Utilizando novamente o exemplo das cadeias de caracteres “Pinot” e “Pinoti”, pela execução do algoritmo é possível observar que através da remoção da última letra “i” da cadeia de caracteres “Pinoti” consegue-se igualar esta a “Pinot”. Logo, a distância entre as duas strings será de 1 caractere. Após isso, é possível obter um grau de similaridade a partir do número de edições necessárias e do tamanho da maior cadeia de caracteres, que no caso seria 6. No exemplo, com uma alteração necessária para transformar “Pinoti” em “Pinot”, o valor para a similaridade seria obtido através de $1 - \frac{1}{6}$, obtendo o valor aproximado de 0,83.

Como levantado em [1], a junção do coeficiente de Sørensen–Dice e do algo-

ritmo de Damerau-Levenshtein pode ser utilizada a fim de obter a similaridade entre cadeias de caracteres. Uma das abordagens consiste em definir um peso para o coeficiente de Sørensen–Dice e outro peso para a saída obtida pelo algoritmo de Damerau-Levenshtein, obtendo como resultado final uma probabilidade de um par de registros ser autêntico ou não. É possível exemplificar novamente com as cadeias de caracteres “Pinot” e “Pinoti”. Considerando um peso de 80% para o coeficiente de Sørensen–Dice e 20% para o algoritmo de Damerau-Levenshtein, utilizando os valores já obtidos para estes, respectivamente de 0,89 e 0,83, obtemos a similaridade ponderada através do cálculo $0,8 * 0,89 + 0,2 * 0,83 = 0,88$. Logo, por conta dos pesos estabelecidos, a similaridade final teria o valor de 0,88.

2.3 SISTEMAS DE RECOMENDAÇÃO

Sistemas de recomendação visam reduzir a sobrecarga da informação, sugerindo um certo item que possa ser considerado de interesse de um usuário. As diferentes abordagens podem ser classificadas como [13]: Filtragem Colaborativa (Seção 2.3.1), Filtragem baseada em Conteúdo (Seção 2.3.2) e Híbrida (Seção 2.3.3).

2.3.1 Filtragem Colaborativa

A filtragem colaborativa visa utilizar dados de outros usuários para realizar as recomendações, com base na ideia de que usuários com histórico parecido, podem ter interesses em comum. Exemplificando, em um site sobre vinhos, se um usuário X e um Y têm históricos similares de vinhos visualizados, quando o X demonstrar interesse em um vinho V ainda não visualizado por Y, pode-se recomendar V para Y, pois ele pode compartilhar esse interesse também.

Popularmente, essa abordagem faz uso de avaliações, e não apenas de histórico de acessos. A ideia é que a filtragem seja realizada com base nas avaliações feitas pelos usuários aos itens, objetivando que usuários com histórico de avaliações similares recebam recomendações de itens bem avaliados pelos outros similares.

É importante ressaltar que, diferentemente da Filtragem baseada em Conteúdo

(Seção 2.3.2), não há necessidade de possuir um conhecimento aprofundado sobre os itens recomendados, e sim ter subsídios para identificar usuários com "gostos" similares. Com isso, pode existir uma diversidade grande na recomendação. Por exemplo, em um site de *e-commerce*, podem ser recomendados ao usuários itens que não tenham características em comum com outros de seu interesse prévio, apenas pelo estudo de comportamento de outros usuários que demonstram interesse por ambos os itens.

Existem duas variações clássicas desse tipo de abordagem de recomendação, que se diferenciam na maneira como estimam a similaridade: se a similaridade é calculada entre usuários (*User-user*) ou entre os itens (*Item-item*) [20]. A primeira possui algumas desvantagens quando se tem usuários ou itens novos, pois ainda não existe um padrão para realizar uma recomendação, e também no quesito de escalabilidade, pois com a base muito grande de usuários, pode ser complicado definir usuários que avaliaram os mesmos itens e calcular as similaridades. A segunda abordagem também tem problemas com itens novos, porém precisa de menos informações de um usuário para recomendar itens similares. Além de ser mais escalável visto que normalmente uma base de itens cresce de maneira mais estável do que uma de usuários.

2.3.2 Filtragem baseada em Conteúdo

Esse tipo de sistema de recomendação, também conhecido como recomendação baseada em conteúdo, usa as características específicas associadas a um item para realizar uma recomendação, utilizando a premissa de que um usuário gostaria de obter recomendações de itens semelhantes (com certas características similares) a itens que ele tenha visualizado ou esteja visualizando. Segundo Lops et al. [17], o processo básico de uma abordagem de recomendação de filtragem baseada em conteúdo consiste em casar os atributos do perfil do usuário, no qual suas preferências e interesses estão armazenados, com os atributos do conteúdo do item, a fim de recomendar ao usuário novos itens que possivelmente possam ser do seu interesse.

Por exemplo, um sistema de recomendação de vinhos pode utilizar características dos vinhos como tipo de uva, vinícola e país. Dessa forma, um usuário pode receber

recomendações de outros vinhos que possuam algumas dessas características em comum com um vinho que esteja visualizando no momento. Outra possibilidade de estratégia de recomendação personalizada seria o usuário receber recomendações de vinhos que compartilhem algumas dessas características com vinhos que fazem parte do seu histórico de vinhos de interesse.

As principais vantagens que esse sistema apresenta são: não precisar de dados de outros usuários; permite selecionar quais características serão utilizadas para gerar recomendações; não favorecer apenas itens mais populares; pode listar facilmente as características que levaram a tal recomendação; conseguir gerar recomendações para usuários com preferências mais únicas. E suas principais desvantagens são: ser difícil encontrar as características apropriadas; não recomendar itens que possam ser de interesse do usuário mas não são similares aos que ele já se interessou ou está visualizando.

2.3.3 Filtragem Híbrida

A partir das abordagens citadas anteriormente (nas Seções 2.3.1 e 2.3.2) é possível gerar a recomendação de itens aos usuários, usando diferentes fontes de dados (histórico, avaliações, descrição dos itens). A filtragem híbrida procura combinar as vantagens que cada uma ofecere. As principais estratégias para combinar as abordagens baseadas em conteúdo e colaborativas são [4]:

- Ponderada: As abordagens de recomendação são implementados separadamente e uma combinação linear dos *scores* é feita para produzir um único *score* final de recomendação.
- Comutação: O sistema utiliza algum critério para alternar entre as abordagens de recomendação. Pode-se também realizar a comutação de uma abordagem nos pontos de desvantagem da outra.
- Mista: As recomendações são geradas pelas abordagens separadamente e são mostradas em uma mesma lista.

- Combinação de *features*: *Features* (características) das diferentes fontes de dados, usadas nas abordagens, são usadas em conjunto em um único algoritmo de recomendação.

- Cascata: O processo de recomendação acontece em fases, de tal maneira que um método é empregado inicialmente (produzindo um resultado intermediário), e em seguida o outro método refina o resultado, gerando as recomendações.

- Aumento de *features*: Combina as abordagens de recomendação usando a saída gerada por uma delas como entrada para a outra.

- Meta-Nível: O modelo inteiro aprendido por uma abordagem se torna entrada para a outra.

2.4 TRABALHOS RELACIONADOS

Diversos trabalhos e estudos abordam tópicos de pesquisa similares aos desse projeto, podendo citar: coleta de dados na Web, pareamento de registros e sistemas de recomendação. O propósito desta seção é descrever alguns destes trabalhos que se relacionam com cada um dos tópicos citados e principalmente utilizados no domínio de aplicação selecionado para este projeto.

2.4.1 Coleta de Dados na Web

No quesito de *Web Scraping*, foram encontradas diversas aplicações. Foram selecionadas duas com maior interseção com a abordagem implementada no presente projeto.

A primeira, apresentada em [14], é implementada usando Node.js[4], como no presente projeto. Esta efetua a coleta de informações sobre produtos genéricos a partir de diversas lojas, ao contrário do presente projeto, em que o foco está na coleta de um tipo específico de produto (vinho). A abordagem implementada por esta

[4]https://nodejs.org/en/

primeira aplicação compreende os seguintes passos: (i) Inicia em uma única página na Web; (ii) O Crawler baixa o HTML da página; (iii) O HTML é analisado para buscar elementos de *links*, como por exemplo toda tag *<a>*, de onde são extraídos os atributos *href* que contém o *link*; (iv) Todos os atributos *href* são adicionados a uma estrutura de dados que é implementada de maneira que não existam duplicatas (evitando uma busca interminável); (v) O próximo item da estrutura é recuperado e o processo volta do começo.

Esse passo-a-passo mostra uma abordagem diferente da utilizada no presente projeto, onde a lista de produtos foi recuperada através de *Sitemaps* e APIs (conforme será apresentado em [14]). Outra diferença foi o SGBD (Sistema de Gerenciamento de Banco de Dados) usado por este, que decidiu seguir com um banco NoSQL, ao contrário do presente projeto que usa um SGBD relacional.

Na segunda aplicação, apresentada em [15], foram utilizadas as bibliotecas *Selenium* e *requests* da linguagem Python, bem populares em aplicações desse tipo. Essa se destaca por utilizar a mesma fonte principal de dados que a deste projeto, o Vivino, para coletar dados de vinhos, vinícolas e avaliações. As informações coletadas de cada uma dessas categorias foram as mesmas, reforçando a importância de cada uma. O que diferencia os projetos é o objetivo final de o que fazer com os dados. Enquanto, no presente projeto, buscou-se parear os vinhos com o de revendedoras, nesta segunda aplicação, buscou-se analisar a percepção dos usuários entre os diferentes tipos e regiões de vinhos, fazendo um estudo de como cada característica do vinho impacta uma avaliação, e tentando criar um modelo preditivo de avaliações.

2.4.2 Pareamento de Registros

Como levantado no Capítulo 1, para o desenvolvimento do presente projeto, foi necessária a implementação de técnicas de *Record Linkage* a fim de validar se um determinado registro de vinho proveniente da plataforma Vivino representa o mesmo vinho oriundo da plataforma de revendedoras. Dentre os trabalhos relacionados sobre *Record Linkage*, é possível citar a abordagem apresentada em [3] visto que possui diversos conceitos acerca de comparabilidade de dados que foram utilizados para a

implementação deste projeto. Tendo como motivação o cálculo de similaridade entre objetos, utilizou-se o domínio de vinho para as experimentações. A utilização de algoritmos de distância de *strings* como Levenshtein também podem ser destacadas como similaridade entre os projetos. A fim de identificar registros com divergências sintáticas, conceitos como blocagem de dados e *record linkage* foram utilizados pelos autores do trabalho.

Outro trabalho que pode ser citado é a ferramenta Reclink [7], onde os autores explicitam a importância de relacionar registros de diferentes bases de dados dentro do setor da saúde do Brasil. Problemas como a ausência de um campo identificador único entre os registros, exibem a necessidade de uma abordagem probabilística, tal como foi executada no presente projeto. Os resultados gerados pela ferramenta Reclink são classificados como pares ou não conforme *score* calculado através de algoritmos fonéticos e de distância de caracteres. No artigo citado, optou-se por estabelecer uma nota de corte para determinado par ser considerado como autêntico ou não.

Um outro trabalho interessante de ser mencionado foi desenvolvido por Mikkel Freltoft Krogsholm, que possui uma ideia similar a deste projeto, que está relatada em seu LinkedIn[5]. Este trabalho faz a coleta e pareamento de informações sobre vinhos obtidas de diferentes fontes. Sua abordagem para coletar os vinhos da revendedora também se aproveita da API interna do site (como explicado em 3.1.1), porém o resultado dessa API foi usado como base principal, diferente desse projeto onde a base principal é a do Vivino. Uma diferença significativa está na estratégia usada para parear os vinhos que foi bem mais simples, sendo realizada pelo nome de cada vinho da revendedora e busca seu nome no Vivino, assumindo que o primeiro resultado é a correspondência correta. Por outro lado, a estratégia abordada no presente projeto é de relizar diversas comparações de diferentes dados, já tendo a base do Vivino previamente coletada (como explicado no Capítulo 4).

[5] http://bit.ly/2ZrKAJU

2.4.3 Sistemas de Recomendação

A abordagem proposta em [21] descreve uma técnica utilizada para extrair informações úteis de um vinho a partir de avaliações feitas por usuários, e utilizar esses dados para criar um sistema de recomendação. A extração é feita através de 5 passos: (i) Normalizar palavras na avaliação de vinhos (remover palavras irrelevantes, pontuação, etc); (ii) Aprimorar o conjunto de palavras normalizadas com multipalavras (bigramas e trigramas); (iii) Padronizar os descritores de vinho em cada avaliação; (iv) Quantificar o cada termo utilizado nas avaliações; (v) Atribuir peso para cada termo, baseado na frequência do termo na base. O último passo tem como saída um conjunto de vetores descritores mapeados ponderados, que podem ser utilizados para criar um sistema de recomendação. Tendo em mãos os descritores de cada vinho da base, ele pode utilizar a abordagem do vizinho mais próximo (que calcula a distância entre vários vetores) para encontrar quais vinhos possuem características próximas. Mesmo se utilizando de avaliações feitas por usuários, o autor apenas as usa para extrair características dos vinhos (avaliações textuais), e utiliza essas características em comum para o sistema de recomendação, então ele está utilizando uma abordagem de Filtragem Baseada em Conteúdo (como explicado na Seção 2.3.2).

Na proposta apresentada em [19], é explorada a quantidade de informação que é necessária para fazer uma boa recomendação de vinhos, comparando os resultados recomendados em duas diferentes bases, onde uma possui um conjunto rico de dados e outra apenas o nome dos vinhos. O conjunto de dados é composto por uma ontologia, contendo as seguintes informações sobre os vinhos: Marca; Região; Cor; Adega; Ano; Castas; Terroir subdividida em clima, solo, terreno, tradição; Sabor subdividida em corpo, sabor, açúcar; Álcool por volume; Preço mínimo; Preço máximo. Com esse conjunto de dados, segue-se para demonstrar como o índice de acerto é maior quando se utiliza a ontologia acima, mas que ela possui um custo no tempo de processamento (visto que os algoritmos precisam lidar com um número maior de dados). Seus experimentos foram feitos utilizando uma Filtragem Baseada em Conteúdo (Seção 2.3.2), onde apenas as características dos vinhos influenciam

a recomendação. No fim, conclui-se que o índice de acerto do algoritmo subiu em 42.97% quando a ontologia foi utilizada, e o custo de processamento dobrou.

Ambas as propostas se assemelham com o presente projeto no quesito de todas usarem uma abordagem de Filtragem Baseada em Conteúdo. O presente projeto buscou propor uma estratégia de recomendação que é feita por uma seleção de vinhos que tenham uma característica básica em comum com o vinho sendo visualizado pelo usuário (Tipo, Vinícola, País) e selecionando aqueles com as maiores notas nas avaliações dos usuários (como explicado na Seção 5.2.4).

3 PROJETO E POPULAÇÃO DA BASE DE DADOS SOBRE VINHOS

Para este projeto, foi desenvolvida uma base de dados para armazenar, de maneira estruturada, informações sobre os vinhos obtidas de diferentes fontes de dados. Tais informações serão coletadas de diferentes fontes de dados na Web. Dessa forma, para permitir a integração de dados, a base também permite armazenar os resultados do pareamento desses registros. As etapas de projeto (Seção 3.1) e população (Seção 3.2) de tal base de dados serão detalhados a seguir.

3.1 PROJETO DA BASE DE DADOS RELACIONAL

Para o projeto dessa base, as seguintes etapas, ilustradas na Figura 5, foram realizadas:

1. **Coleta**: Obter dos dados sobre vinhos de cada uma das fontes
2. **Análise**: Analisar os dados que podem ser obtidos de cada uma das fontes, incluindo a descoberta sobre quais são comuns entre as fontes (para que possam ser usados posteriormente para o pareamento de registros, discutido no Capítulo 4)
3. **Seleção**: Selecionar quais dados são mais importantes para a comparação dos vinhos e que deverão ser armazenados na base
4. **Modelagem conceitual**: criação do modelo conceitual da base de dados
5. **Modelagem lógica**: criação do modelo lógico da base de dados
6. **Modelagem física**: criação do modelo físico da base de dados

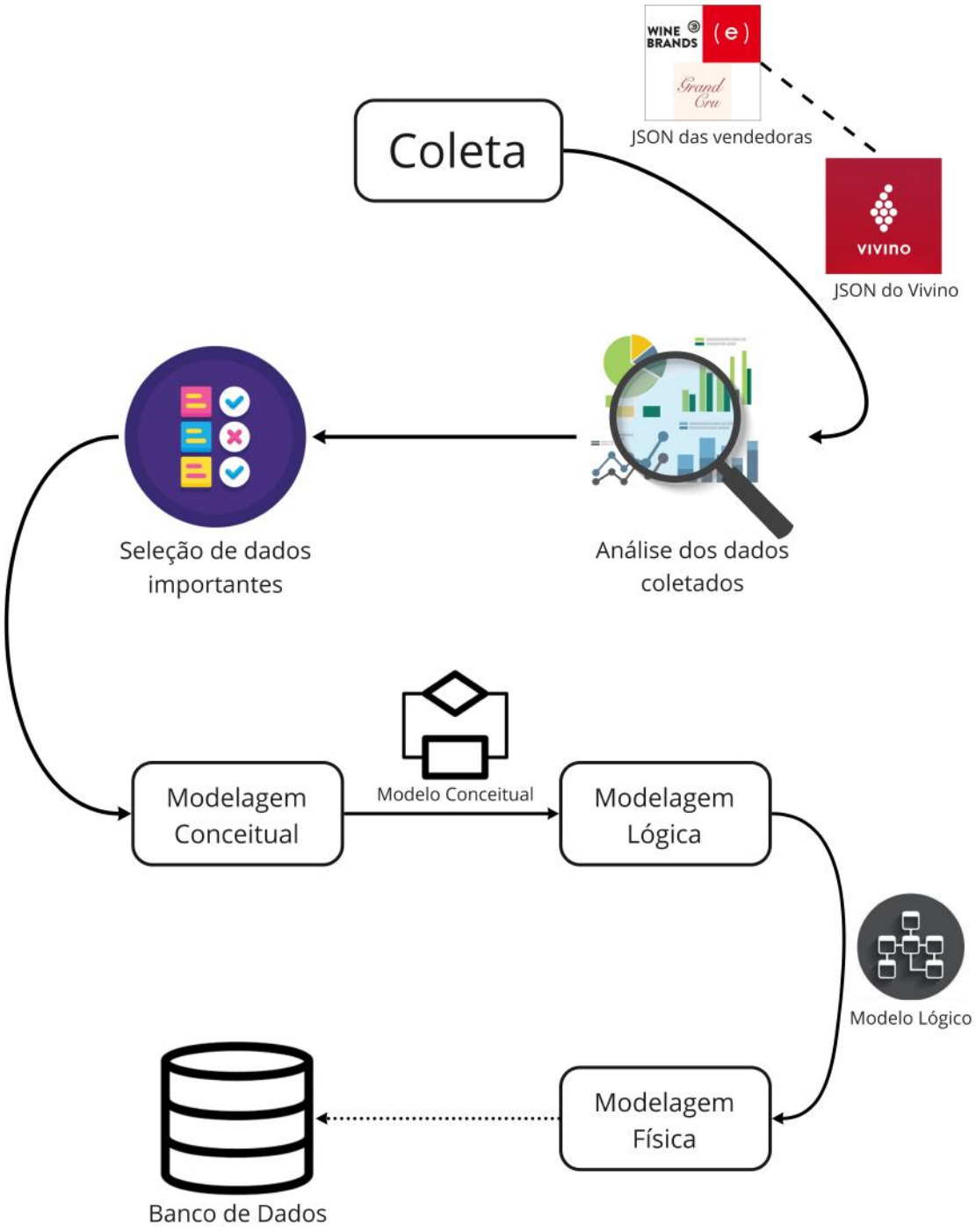

Figura 5: Principais etapas do projeto da base de dados.

Nas próximas seções serão detalhadas como cada uma dessas etapas foi desenvolvida. Na Seção 3.1.1, são explicadas as etapas 1 e 2 na Seção 3.1.2, a etapa 3 e, na Seção 3.1.3, as etapas 4, 5 e 6.

3.1.1 Coleta e Análise

O primeiro passo foi definir quais sites sobre avaliação e venda de vinhos seriam utilizados no presente projeto. Para isso, foram estudados diversos sites de avaliação e venda de vinhos, a fim de encontrar os mais populares e que possuíssem os dados de maneira estruturada. Ao final deste estudo, o site contendo avaliação sobre vinhos escolhido foi o Vivino (Seção 3.1.1.1) e os sites de revendedoras selecionados foram o Winebrands (3.1.1.2), o Grand Cru (3.1.1.3) e o Evino (3.1.1.4).

Para realizar a coleta dos dados, foi necessário estudar diversos mecanismos que possibilitassem obter os dados de um site. Foram usados diversos métodos, pois cada site possuía um jeito diferente de retornar as informações do *back end*[1] para o *front end*[2]. Nas próximas seções (3.1.1.1, 3.1.1.2, 3.1.1.3 e 3.1.1.4) serão detalhados esses mecanismos.

3.1.1.1 Vivino

O Vivino possui um mapa do site em XML que lista as URLs das páginas dos vinhos em seu catálogo. Embora este *sitemap* não estivesse cobrindo 100% dos vinhos do catálogo (não estava totalmente atualizado), este foi utilizado como base para a coleta realizada no presente projeto[3].

Sendo assim, o primeiro passo foi baixar todos os arquivos XML que possuíam URLs de páginas detalhando algum vinho. Depois, foi feita uma conversão para uma lista em formato JSON, facilitando a utilização posterior (visto que esse formato de dados também era usado pelas revendedoras).

No início do desenvolvimento da coleta de dados neste projeto, em setembro

[1] *Back End* é a camada que funciona "por trás" de uma aplicação, responsável por alimentar os dados a serem usados pelo *Front End*.

[2] *Front End* é o termo utilizado para designar a parte visual de um site com a qual o usuário interage.

[3] Ao longo do estudo e coleta desses dados, foi realizado contato com o suporte do Vivino, através de seu CTO (*Chief Technical Officer*), para tirar algumas dúvidas e explicando o objetivo de baixar os dados sobre os vinhos para a realização do presente projeto.

de 2019, o Vivino possuía uma API REST, na qual era possível, a partir de um identificador de um vinho (que é informação apresentada na URL da página do vinho, podendo ser obtida da lista do *sitemap*), receber um JSON com todas as informações que eram necessárias. Porém, essa API não se encontra mais disponível, e foi necessária uma nova abordagem para efetuar a coleta dos dados.

Foi feito um estudo para entender melhor como o *front end* do Vivino recebia essas informações e foi encontrado, no HTML de cada vinho, um JSON, que era o mesmo retornado através da API anteriormente disponível. Assim, foi feito um laço que percorre a lista de URLs, busca o conteúdo HTML de cada página e retorna esse JSON. É possível verificar um exemplo de JSON contendo os dados de um vinho proveniente do Vivino através do Apêndice B.

3.1.1.2 Winebrands

Assim como o Vivino, o Winebrands também possui um *sitemap* com a lista de *links* para todos os vinhos de sua base. Porém, frequentemente, este arquivo ficava indisponível, além de aparentar estar desatualizado pois possuía muitos *links* quebrados. Então, foi necessário encontrar outro meio de obter a lista de vinhos. Para isso, foi utilizada a biblioteca Puppeteer[4] (uma biblioteca do Node.js que fornece uma API de alto nível para controlar o Chrome ou Chromium "headless" através do protocolo DevTools[5]).

A seção de buscas do site do Winebrands lista todos os vinhos que correspondem a um filtro fornecido. Com a ajuda do Puppeteer, é possível acessar a busca, passando apenas os filtros de número da página e limite de vinhos por página e, após ela ser carregada, extrair todos os identificadores de vinhos presentes. Assim, basta fazer uma rotina que se repete até a busca não retornar mais nenhum elemento, acabando assim com uma lista onde cada elemento é um identificador. Após isso, é necessário buscar o JSON de cada vinho, que é possível fazendo uma requisição HTTP usando o método GET e passando o identificador do vinho (informação presente no JSON)

[4] https://pptr.dev/

[5] https://developers.google.com/web/tools/puppeteer

como parâmetro. É possível verificar um exemplo de JSON contendo dados de um vinho proveniente do Winebrands através do Apêndice C.

3.1.1.3 Grand Cru

Similar ao método usado no Winebrands, no site do Grand Cru, também foi necessário fazer um laço para realizar requisições passando como parâmetro o número da página e o limite de vinhos por página. A diferença em relação ao Winebrands é que no Grand Cru não eram retornados apenas os identificadores de cada vinho, mas sim o JSON completo de cada um, ou seja, no final de todas as iterações do laço, já estava disponível uma lista com todas as informações necessárias. É possível verificar um exemplo de JSON contendo dados de um vinho provenientes do Grand Cru através do Apêndice D.

3.1.1.4 Evino

A coleta de dados a partir do site do Evino foi similar a do Vivino, ele também possui um *sitemap* com todos os vinhos de sua base. Para pegar o JSON de cada um também seria necessário fazer uma requisição em cada página. A diferença principal foi que, nesse caso, também foi usado o Puppeteer, visto que o JSON não estava presente no HTML da página que nem no Vivino, mas sim em um *script* em *JavaScript* de cada página. A biblioteca possibilitou acessar cada página e, dentro dela, acessar a camada de dados (*dataLayer*), que possuía o objeto completo. É possível verificar um exemplo de JSON contendo dados de um vinho provenientes do Evino através do Apêndice E.

3.1.2 Seleção

A etapa de seleção de dados que tem como objetivo, após a análise dos dados recebidos, selecionar os dados importantes de serem incluídos no projeto do banco de dados, com atenção especial a incluir os dados comuns recebidos de cada fonte (que serão fundamentais para a estratégia de pareamento de registros). Visto que

este projeto precisa integrar informações de quatro bases diferentes, essa etapa se torna essencial.

Após analisar os resultados que seriam obtidos de cada fonte, foram selecionadas as seguintes informações em comum, consideradas mais importantes para possivelmente serem usadas no pareamento de registros (apresentado no Capítulo 4):

- Nome do vinho
- Vinícola a qual esse vinho pertence
- Tipo (Tinto, Branco, Rose, Espumante, etc.)
- País de origem do vinho
- Região do país onde o vinho foi produzido

Estas foram consideradas as informações principais a serem armazenadas sobre cada vinho na base de dados. Além disso, informações sobre a partir de qual fonte de dados os registros que serão armazenados foram obtidos são necessárias, bem como as informações pertinentes ao pareamento de registros em si (permitindo indicar os registros pareados das diferentes fontes). O projeto em nível conceitual, lógico e físico da base de dados será explicitado na seção seguinte (Seção 3.1.3).

3.1.3 Modelagem Conceitual, Lógica e Física da Base de Dados

Neste projeto, o pareamento de registros objetiva parear os registros obtidos do Vivino com os das demais revendedoras. Isto foi pensado porque o site do Vivino possui informações importantes sobre avaliações de usuários (que as demais fontes não possuem), por outro lado, as revendedoras podem fornecer detalhes mais específicos sobre os vinhos e até possibilitar que os mesmos sejam adquiridos através de seus sites de *e-commerce*. Com isso, é necessário entender quais informações são mais importantes de se ter a respeito de um vinho, além de quais mais se repetem, com a finalidade de atingir uma estrutura de tabelas mais normalizada (para padronizar os dados e evitar replicações). Dessa forma, foram criadas as entidades

com informações comuns tanto ao Vivino quanto a qualquer revendedora, uma que conecta os dois, e outra mais específicas de cada um.

O diagrama ER (Entidade-Relacionamento) [12] apresentado na Figura 6 ilustra o modelo em nível conceitual proposto. Os conceitos centrais modelados são focados em descrever os vinhos do Vivino e das revendedoras, existindo 2 entidades separadas para eles. As demais entidades servem para caracterizar pelo menos uma dessas duas. Tanto no Vivino quanto nas revendedoras, existiam as informações básicas de:

- Qual a vinícola desse vinho.
- Qual o tipo desse vinho (ex.: Tinto/Branco/Rose/Espumante/Outros).
- De onde vem esse vinho (País/Região).

Cada uma dessas informações foi modelada como uma entidade para que elas fiquem normalizadas e relacionadas a descrever os vinhos. Outras informações foram previstas na modelagem, mesmo que só existissem para o Vivino, que são uvas, estilo do vinho, número de avaliações e nota média das avaliações. Além disso, os registros dos vinhos das revendedoras estão relacionados a qual revendedora foram obtidos.

Por fim, temos a relação entre um vinho do Vivino e da revendedora, para armazenar informações sobre o pareamento de registros entre estas fontes. Nesse relacionamento, é previsto um atributo para guardar um *score* de similaridade entre nome da vinícola e também do nome do vinho (conforme detalhado no Capítulo 4). Tal como discutido na Seção 2.2.2, este cenário favorece a utilização de técnicas de *Record Linkage*, mais especificamente de maneira probabilística, visto que registros representando o mesmo vinho podem conter alguma diferença de digitação e informação entre as fontes de dados a partir das quais foram obtidas.

O modelo lógico está apresentado na Figura 7 e foi projeto para um banco de dados relacional, com base no modelo conceitual previamente apresentado. A organização da Figura 7 busca estar próxima da organização da Figura 6, previamente apresentada contendo o modelo conceitual, para que seja mais fácil comparar os dois

modelos. O dicionário de dados pode ser encontrado no Apêndice A. O projeto físico do banco de dados foi desenvolvido usando o SGBD (Sistema de Gerenciamento de Banco de Dados) relacional MySQL[6] (em sua versão 5.7).

[6]https://www.mysql.com/

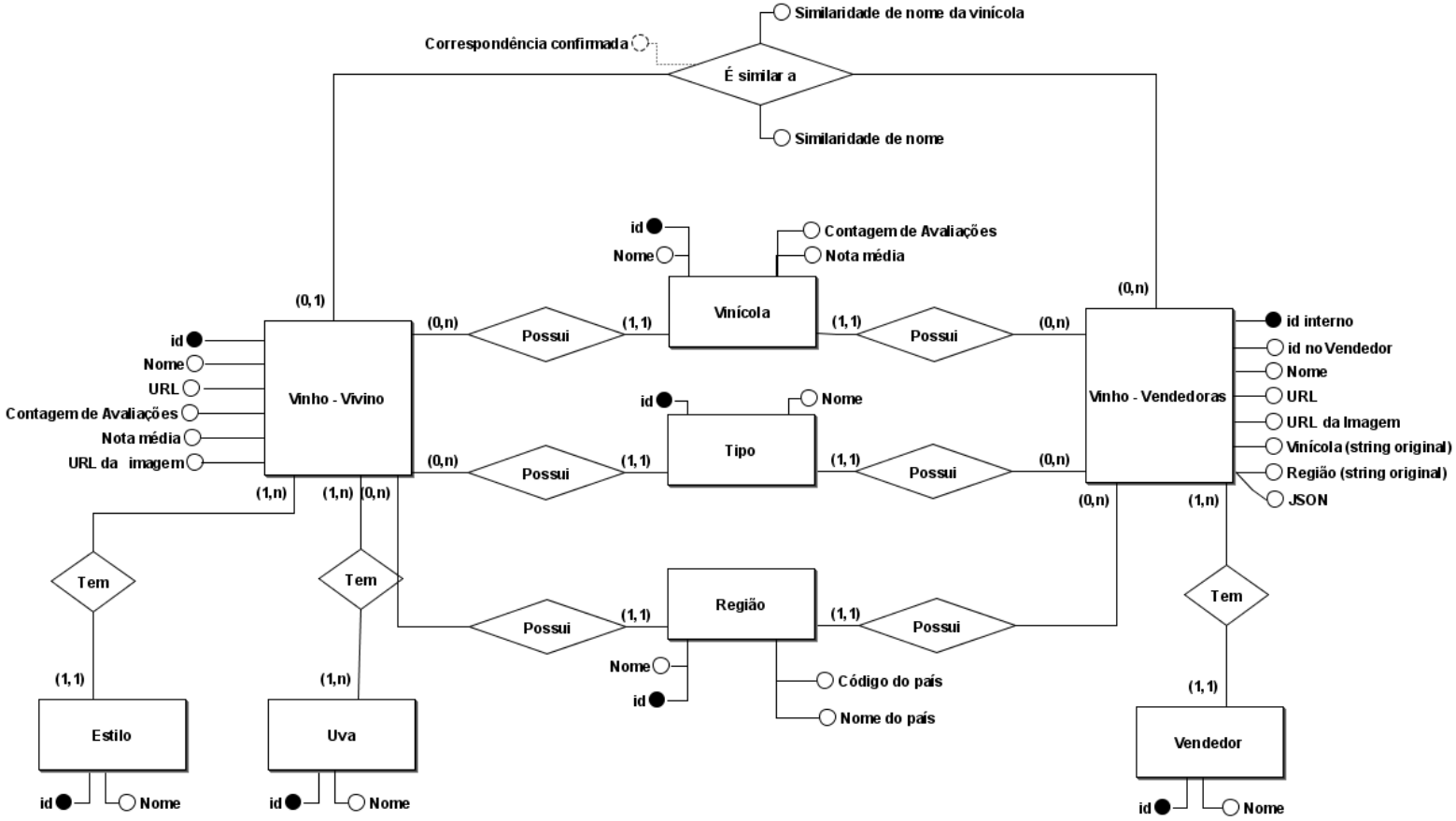

Figura 6: Modelo conceitual.

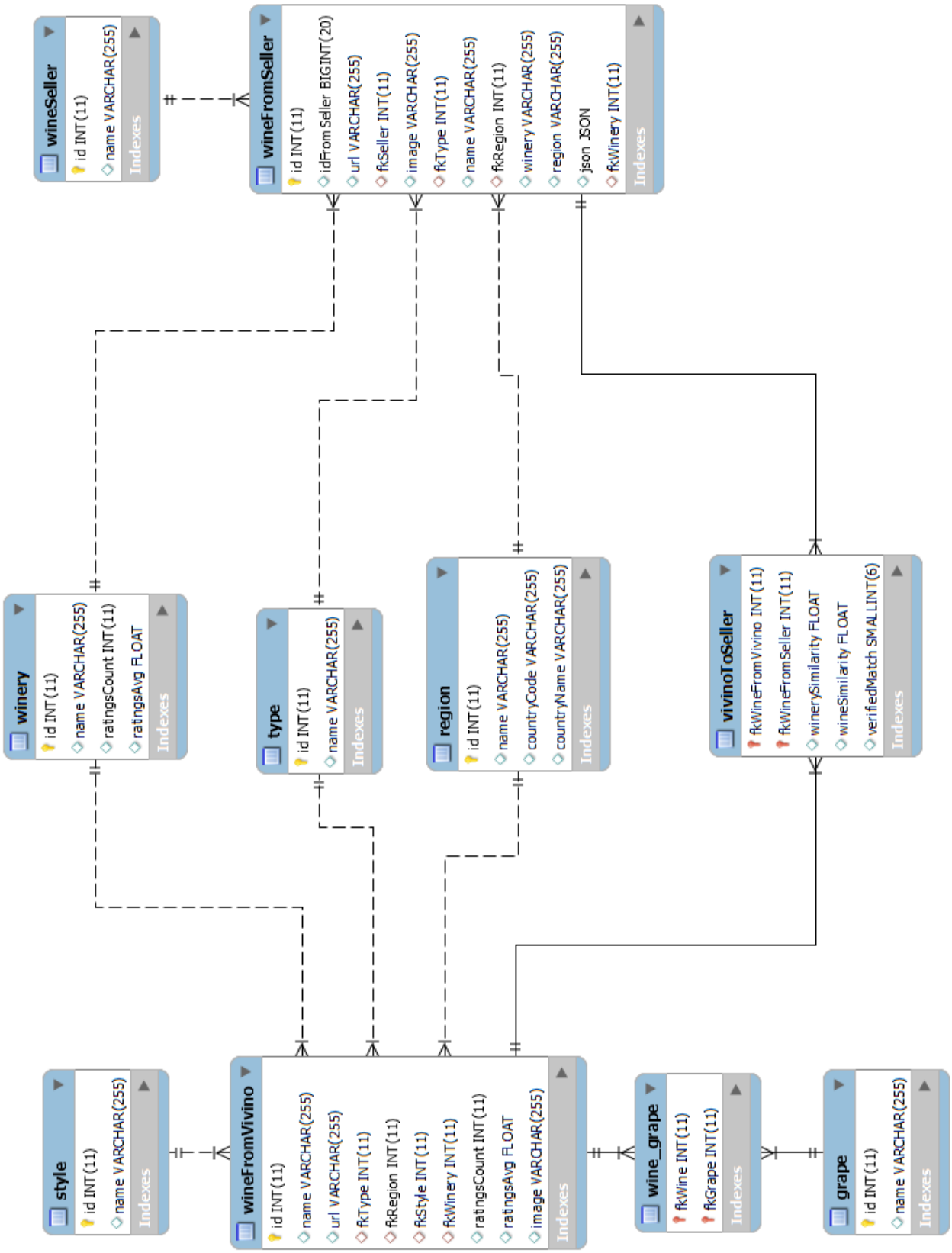

Figura 7: Modelo Lógico.

3.2 POPULAÇÃO DA BASE DE DADOS

Para facilitar a automatização da população do banco de dados, a partir dos dados coletados das diferentes fontes de dados, foram feitos alguns ajustes. Estes objetivavam padronizar o formato dos arquivos JSON obtidos e já normalizar os conteúdos de diferentes registros.

No caso do Vivino, que possui mais informações, foi feito um *parser* específico. Algumas das principais etapas estão listadas abaixo:

- Estruturação do JSON do Vivino para um mais simples e somente com as informações que serão usadas no presente projeto.
- De-Para do tipo, que vem como um identificador (para isso foi necessário buscar esse "de-para" no Vivino)

Abaixo pode ser encontrado o código JSON final do Vivino:

Código 3.1: JSON formatado com informações do Vivino

```
{
  "id": 2895189,
  "name": "Moscatel Rose",
  "url": "URL do vinho no Vivino",
  "type": "Sparkling",
  "region": {
    "id": 1465,
    "name": "Serra Gaucha",
    "countryCode": "br",
    "countryName": "Brazil"
  },
  "style": {
    "id": 253,
    "name": "Brazilian Sparkling"
  },
  "winery": {
    "id": 29186,
    "name": "Garibaldi",
    "ratingsCount": 26220,
    "ratingsAvg": 3.7
  },
  "ratingsCount": 981,
  "ratingsAvg": 4.1,
  "image": "URL da imagem do vinho no Vivino",
  "grapes": [
    {
      "id": 1739,
      "name": "Moscatel"
    }
  ]
}
```

Após a estruturação desse JSON, a rotina de coleta foi executada para popular a base com os vinhos do Vivino. Os vinhos das revendedoras foram populados no final da rotina de pareamento (conforme apresentado no Capítulo 4).

4 PAREAMENTO DE REGISTROS SOBRE VINHOS

Considerando a base de dados descrita no Capítulo 3, foi necessário elaborar e implementar estratégias para o pareamento de registros, que foram implementadas em Node.js. No presente projeto, o objetivo é comparar os vinhos obtidos a partir do site do Vivino (que já estão coletados e disponíveis na base) com os obtidos a partir dos sites das revendedoras, buscando estabelecer a relação de correspondência entre eles (pareamento de registros).

O pareamento de registros desenvolvido no presente projeto segue as cinco etapas principais, previstas na abordagem geral proposta por Peter Christen [5]. A Figura 8 ilustra o fluxo de realização destas etapas.

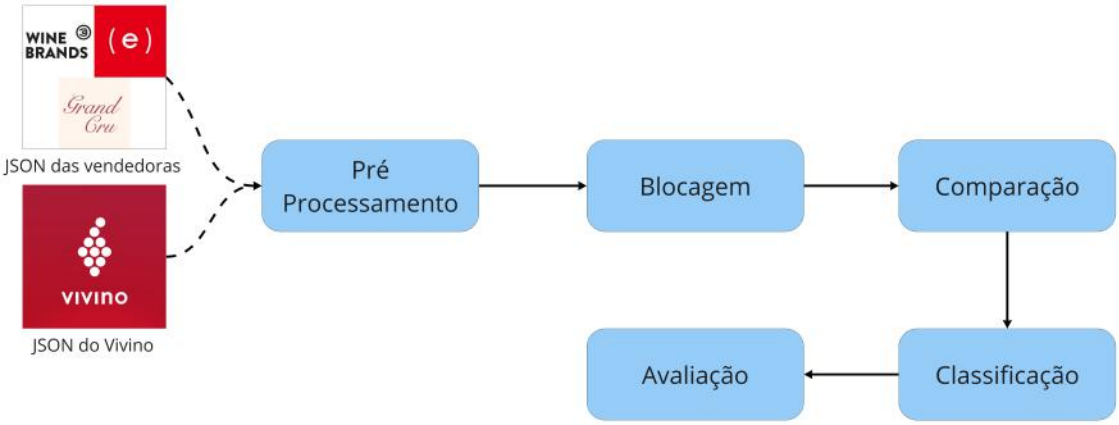

Figura 8: Principais etapas do processo de pareamento de registros.

Nas seções a seguir, as estratégias adotadas para o desenvolvimento de cada uma das etapas serão detalhadas. Inicialmente, a etapa de Pré-processamento (Seção 4.1) é responsável pela limpeza e padronização dos dados, os ajustes principais previram padronização de conteúdo e idioma. A seguir, a etapa de Blocagem (Seção 4.2) consiste em blocar os dados a fim de reduzir o número de registros que serão efetivamente comparados. Na terceira etapa, a de Comparação (Seção 4.3), são escolhidos os campos a serem comparados e diferentes técnicas são aplicadas para cálculo de similaridade, como, por exemplo, o coeficiente de Sørensen–Dice e o algoritmo de Damerau-Levenshtein. Na quarta etapa, a de Classificação (Seção 4.3), com base na

saída da etapa anterior, a classificação de determinado par de registros ser autêntico ou não pode ser definida usando limiares (*thresholds*). Por fim, na etapa de Avaliação (Seção 4.4), foi realizado um processo de avaliação manual a fim de avaliar a precisão do processo.

4.1 PRÉ-PROCESSAMENTO

O objetivo do pareamento de registros proposto é possibilitar identificar pares de registros que representem o mesmo vinho, sendo este par composto por um registro obtido de uma revendedora e o outro do Vivino. Porém, cada site possui sua própria estrutura de dados, modo de escrita, de formatação, entre outros. Isso dificulta muito a tarefa de achar uma correspondência. Esta seção busca ilustrar como foi implementada a etapa de pré-processamento, importante para preparar os dados para as etapas subsequentes.

Um dos problemas enfrentados para padronização e obtenção de dados, foi que a plataforma Vivino, utilizada como base de dados primária para os vinhos (os vinhos das revendedoras serão pareados exclusivamente com os do Vivino), armazena suas informações utilizando a língua inglesa. Como as revendedoras utilizadas (Winebrands, Grand Cru e Evino) são nacionais, informações como o país do vinho estavam na língua portuguesa.

Para que pudesse ser feita a comparação entre os nomes dos países de origem dos vinhos da plataforma Vivino com as demais plataformas, foi necessária a utilização de um pacote externo responsável por traduzir textos de português para inglês, o *google-translate-open-api*[1]. Com a utilização do *npm*[2], a importação do pacote foi feita de maneira simples e prática.

Porém, mesmo com a padronização idiomática, ainda não era possível garantir que a comparação entre duas *strings* seria exata. A razão deste problema é a utilização de mais de um espaço em branco em sequência, traços, letras maiúsculas e

[1] https://www.npmjs.com/package/google-translate-open-api
[2] https://nodejs.org/en/knowledge/getting-started/npm/what-is-npm/

minúsculas, além de alguns países possuírem seu nome oficial e completo, por exemplo *Federative Republic of Brazil* (nome oficial do país) será tratado apenas como *Brazil* (versão simplificada do nome do país). Para permitir essas comparações, também foi necessária uma padronização neste sentido.

Buscando simplificar o desenvolvimento desta etapa, foi criado um serviço para cada revendedora que é responsável por estruturar os dados de cada vinho e buscar o dado dentro da estrutura do JSON de cada revendedora. Dentro desses serviços, é feito um laço que percorre seus vinhos, e para cada um, é chamada a função exibida no Apêndice F. Esse trecho é responsável por chamar cada função do serviço, e também da classe que faz os acessos ao banco de dados.

Aproveitou-se o polimorfismo do JavaScript para passar a classe do serviço como um parâmetro para essa função, que é objeto *seller*. O objeto *wine* é o JSON de cada vinho. Os nomes atribuídos foram escolhidos de modo a tornar o código sempre o mais legível possível. Caso qualquer erro ocorresse durante a execução, a saída ficaria como *null*, para ser facilmente tratável após a chamada da função.

A grande vantagem do método *getWineObject*, que será explicado a seguir, é a utilização do conceito de *polimorfismo*, ou seja, classes que têm a mesma identificação, porém um comportamento distinto. Por exemplo, as classes *evino*, *grandcru* e *winebrands* representam o mesmo conceito, uma revendedora de vinhos. Porém, cada uma dessas classes têm um comportamento diferente em sua implementação, devido às diferentes regras de negócio aplicadas pelos sistemas das revendedoras, e dos diferentes formatos da resposta de cada uma, que geravam a necessidade de fazer um "De-Para" do dado da revendedora para o formato esperado, e também uma higienização. Os códigos completos responsáveis por fazer os De-Paras e pré-processamento do JSON como veio da revendedora para o formato esperado pode ser encontrado no Apêndice H.

Um conceito muito importante utilizado para garantir o funcionamento do código apresentado no Apêndice F, é o *Duck Typing*. Por conta do JavaScript ser uma linguagem de tipagem dinâmica e não possuir o conceito de interfaces, é necessário garantir que as classes *evino*, *grandcru* e *winebrands* representem o mesmo conceito

dentro do sistema, apenas com implementações diferentes.

Se um objeto possui métodos e propriedades como *getWineCountry*, *getWineName* e *isWine*, esse objeto é uma representação de uma revendedora de vinhos. Nos exemplos a seguir, apresentados nos Códigos 4.1, 4.2 e 4.3, apenas em tempo de execução será decidido qual implementação do método *getWineName* será chamada, e cada uma irá fazer o "De-Para" e Higienização do Nome de acordo com a revendedora relacionada.

Código 4.1: Implementação do método *getWineName* para o Evino

```
function getWineName(json) {
    let name = _.get(json, 'productName', _.get(json, 'productDetail.
        name', ''));
    name = name.replace(this.getWineYear(json), '');
    return name;
}
```

No caso da plataforma Evino, os arquivos *JSON* contêm uma chave chamada *productDetail.name*, responsável por guardar o nome do produto.

Código 4.2: Implementação do método *getWineName* para o Grand Cru

```
function getWineName(json) {
    let name = _.get(json, 'title', _.get(json, 'variants[0].name', '')
        );
    name = name.replace(`${this.getWineYear(json)}`, '').trim();
    name = name.replace(`${this.getWineBottleSize(json)}`, '').trim();
    return name;
}
```

Para o Grand Cru, a proposta é similar ao Evino. O nome pode ser obtido através do campo *title*. Após isso, quaisquer outras informações como ano e tamanho da garrafa são descartadas.

Código 4.3: Implementação do método *getWineName* para o Winebrands

```
function getWineName(json) {
```

```
    const name = json.productName;
    return _.trim(name.split(' - ')[1].replace(this.getWineWinery(json)
        , ''));
}
```

O nome do vinho no site Winebrands pode ser obtido através da extração do campo *productName*. Após isso, quaisquer informações presentes acerca da vinícola neste campo são removidas.

Além dos tratamentos mencionados, foi feita também uma tratativa para qualquer texto que fosse ser comparado na etapa 4.3 (e pode ser visto no Código 4.7), que envolve:

- Remover espaços duplos (*replaceCaseInsensitive*)
- Remover qualquer caracter especial (*removeDiacritics*)
- Transformar todos os caracteres em letra minúscula.

4.2 BLOCAGEM

A abordagem mais simples seria comparar todos os dados de cada vinho de uma revendedora, com todos os outros do Vivino. Porém, isso não é uma tarefa muito performática, especialmente para realizar um MVP (Mínimo Produto Viável) como foi feito para esse projeto, visto que o Vivino tem mais de 1 milhão de vinhos coletados e cada revendedora tem em média aproximadamente 500 vinhos. Isso seriam mais de 500 milhões de comparações entre registros de vinhos (sendo que a comparação para o pareamento de registros pode envolver diferentes campos) para cada revendedora.

Sendo assim, foi necessário reduzir o número de vinhos do Vivino a serem comparados, pois é necessário olhar 100% dos vinhos das revendedoras. Então, a abordagem escolhida foi buscar reduzir o escopo de vinhos começando com dados mais generalistas (como, por ex., país, tipo, etc.). Visando essa redução, os passos adotados foram:

- Isolar as vinícolas do Vivino que sejam do mesmo país do vinho atual da revendedora.
  - Isso pode ser feito com uma comparação simples, visto que esses dados já foram normalizados (como descrito na Seção 4.1).
- Isolar os vinhos da mesma vinícola
  - Para encontrar qual é a vinícola do Vivino correspondente, foi necessário comparar o nome de cada uma, de acordo com a abordagem explicada na seção seguinte (Seção 4.3).
- Isolar os vinhos do mesmo tipo
  - Isso também pode ser feito com uma comparação simples, visto que esses dados já foram normalizados (como descrito na Seção 4.1).
- Comparar todos os vinhos da mesma vinícola com esse tipo
  - Utilizando também a abordagem explicada na seção seguinte (Seção 4.3).

4.3 COMPARAÇÃO E CLASSIFICAÇÃO

Conforme discutido previamente no Capítulo 2, existem diversos algoritmos para comparar duas *strings* (cadeias de caracteres). Por conta dos critérios levantados na Seção 2.2.2, foi utilizado o algoritmo de Damerau–Levenshtein em conjunto do coeficiente de Sørensen–Dice. Porém, os testes apontaram indícios de que seria necessário elaborar uma estratégia que gerasse um *score* único, onde este seria obtido usando uma média ponderada combinando o algoritmo de Damerau-Levenshtein e o coeficiente de Sørensen–Dice. Após algumas rodadas de validação manual (sendo esta melhor descrita em 4.4), o melhor resultado obtido levando em consideração os *matches* estabelecidos pela execução dos algoritmos foi definir um peso de 80% para o coeficiente de Sørensen–Dice e 20% para a saída do algoritmo de Damerau-Levenshtein.

Algumas das revendedoras possuíam originalmente em suas bases produtos que não são vinhos, como kits, taças e sucos. Para isso, foi necessário que se implementasse os Códigos 4.4, 4.5 e 4.6, permitindo que o pré-processamento já filtre objetos que não forem um vinho. Abaixo, é possível ver as diferentes implementações necessárias para definir se determinado objeto é de fato um vinho.

Código 4.4: Implementação do método *isWine* para o Evino

```
function isWine(json) {
    return _.get(json, 'productDetail.classification', '') === 'Wine';
}
```

No caso da plataforma Evino, os arquivos *JSON* contêm uma chave chamada *productDetail.classification*, responsável por classificar o tipo do produto como vinho ou qualquer outro produto.

Código 4.5: Implementação do método *isWine* para o Grand Cru

```
function isWine(wine) {
    return true;
}
```

Para o Grand Cru, todos os produtos obtidos sempre serão apenas vinhos, porém, para manter a utilização do polimorfismo e reutilização de código, o método *isWine* foi implementado para garantir que as chamadas pudessem ser feitas sem problemas, assim como visto no código do Apêndice F.

Código 4.6: Implementação do método *isWine* para o Winebrands

```
function isWine(json) {
    return _.get(json, 'productName', '').startsWith('Vinho');
}
```

O site do Winebrands possui uma abordagem peculiar. Como não há classificação do produto por conta da plataforma, foi necessário extrair o nome de cada produto obtido e validar se o mesmo começa com a palavra “Vinho”.

Apenas utilizar os algoritmos não se mostrou suficiente, foi necessário também realizar alguns tratamentos adicionais nas duas strings para remover palavras muito similares. Por exemplo, ao comparar dois vinhos, o Vivino pode listar o nome como "Willamette Valley Pinot Noir" enquanto uma das revendedoras pode apenas listar como "Willamette Valley" e deixar o "Pinot Noir" apenas como tipo do vinho. Se compararmos apenas as 2 strings diretamente, a similaridade delas iria ser prejudicada por causa da falta do "Pinot Noir" enquanto algum outro vinho (exemplo "West Valley Pinot Noir") poderia ser beneficiado por trazer essas palavras no nome, por ter uma similaridade maior de acordo com os algoritmos. Por causa disso, foi utilizada uma rotina que contava a frequência de palavras tanto no nome do vinho e da vinícola, feito um estudo manual sobre o que significa cada palavra que estivesse presente em mais de 50% da base, e finalmente antes de comparar as duas *string*, são removidas essas palavras de cada *string* sendo comparada.

Em alguns casos, vinhos provenientes do Vivino e das revendedoras possuíam informações incompletas acerca da vinícola. Para evitar de deixar as informações em branco, as plataformas em grande parte dos casos replicavam o nome do vinho no campo referente à vinícola. Um exemplo pode ser visualizado na Figura 9, onde a Fonte B exemplifica o problema apresentado.

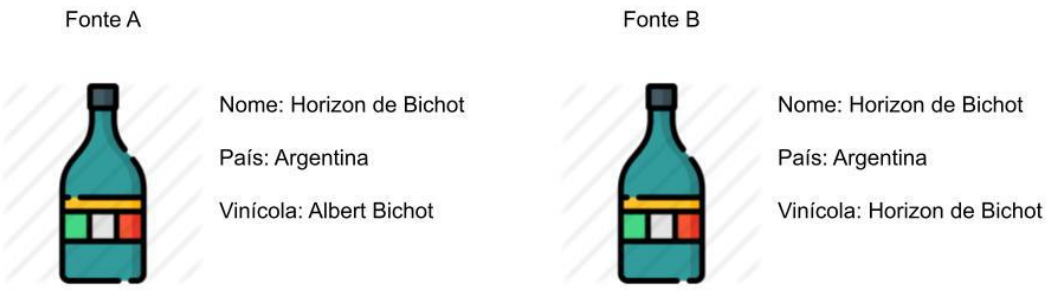

Figura 9: Fonte B em um cenário de vinícola errada.

Como esse problema foi encontrado em diferentes vinhos e vinícolas, o algoritmo precisou ser adaptado e ser adicionada uma etapa extra no comparativo da vinícola, onde, ao invés de comparar apenas a vinícola da revendedora com as vinícolas do Vivino, foi comparado também o nome do vinho da revendedora com as vinícolas do Vivino a fim de encontrar a correspondência correta da vinícola. Citando novamente o exemplo da Figura 9, ao comparar diretamente o valor "Albert Bichot" da Fonte A com o valor "Horizon de Bichot" da Fonte B, não seria possível estabelecer uma igualdade. Porém, com a comparação entre o nome do vinho da Fonte A ("Horizon de Bichot") e o nome da vinícola da Fonte B (também "Horizon de Bichot"), foi possível evitar o problema.

Foi necessário desenvolver uma solução que resolvesse esses problemas previamente citados, o Código 4.7 ilustra a função principal para o cálculo de similaridade entre duas *strings* (também implementa o pré-processamento explicado na Seção 4.1), calcular um *score* de similaridade, utilizando bibliotecas[3][4] para o cálculo da distância de Damerau–Levenshtein e do coeficiente de Sørensen–Dice, com seus respectivos pesos.

Código 4.7: Função que calcula a similaridade entre 2 strings

```
function getMatchSimilarity(_str1, _str2) {
    const str1 = _.toLower(removeDiacritics(replaceCaseInsensitive(
       _str1, ' ', '')));
    const str2 = _.toLower(removeDiacritics(replaceCaseInsensitive(
       _str2, ' ', '')));
    const diceSim = dice(str1, str2);
    const lvSim = lv(str1, str2).similarity;
    const diceWeight = 0.8;
    const lvWeight = 1 - diceWeight;
    return {
        dice: diceSim,
        lv: lvSim,
        score: (lvSim * lvWeight) + (diceSim * diceWeight),
    };
```

[3]https://www.npmjs.com/package/clj-fuzzy
[4]https://www.npmjs.com/package/damerau-levenshtein

```
}
```

Após, seguem-se as comparações entre os nomes das vinícolas e vinhos. O Código 4.8 implementa estas comparações e a etapa de classificação, vale ressaltar que não foi selecionado um *score* mínimo para a classificação, foi salvo o *match* que tivesse a melhor nota, para fins de análises futuras.

Código 4.8: Função que calcula similaridade das vinícolas

```
// ... Código para selecionar todas as vinícolas do Vivino
wineries.forEach((w) => {
    matchWineryName.push({ ...w,
        score: _helpers.getMatchSimilarity(
            removeCommonWordsFromWineryName(w.name),
            removeCommonWordsFromWineryName(winery)),
        vivinoWineryForMatch: removeCommonWordsFromWineryName(w.name),
        sellerWineryForMatch: removeCommonWordsFromWineryName(winery),
        winery });
    matchWineName.push({ ...w,
        vivinoWineryForMatch: removeCommonWordsFromWineryName(w.name),
        sellerWineForMatch: wineName,
        score: _helpers.getMatchSimilarity(
            removeCommonWordsFromWineryName(w.name), wineName),
        winery });
});
matchWineryName = _.chain(matchWineryName).orderBy((e) => e.score.score
    , ['desc']).first()
     .value();
matchWineName = _.chain(matchWineName).orderBy((e) => e.score.score, ['
    desc']).first()
     .value();
if (matchWineryName.score >= matchWineName.score) { return
    matchWineryName; }
return matchWineName;
```

Neste código (Código 4.8), o algoritmo percorre todas as vinícolas do Vivino, chama as funções que removem palavras comuns, e calcula a similaridade entre

cada vinícola do Vivino com a atual da revendedora, e também a similaridade do nome do vinho (removendo as palavras comuns de vinho) da revendedora com a vinícola do vinho. O motivo de comparar o nome do vinho com o nome da vinícola é identificar casos onde a vinícola estava cadastrada, erroneamente, com o mesmo nome do vinho (como exemplificado na Figura 9). No final, compara-se a nota retornada pelo algoritmo usado no nome do vinho e no nome da vinícola e retorna o que tiver maior nota. A saída desta etapa diz qual vinícola do Vivino é par da vinícola do vinho atual da revendedora.

Já possuindo a vinícola correspondente, são selecionados todos os vinhos da base que pertencem a ela, e então é executado o Código 4.9, responsável por percorrer todos os vinhos do Vivino que são dessa vinícola, e para cada um deles:

- Chamar as funções que removem palavras comuns do nome do vinho.
- Calcular a similaridade entre as duas palavras.
- Retornar esses *scores* de similaridade.
- Selecionar o vinho do Vivino que tiver o maior *score*, e considerar que esse vinho é a correspondência que estava sendo procurada.

Código 4.9: Função que calcula similaridade dos vinhos

```
vivino.forEach((r) => {
    match.push({
        ...r,
        wineName: name,
        wineNameWithoutWinery: nameReplaced,
        vivinoNameForMatch: removeCommonWordsForWineMatch(r.name),
        sellerNameForMatch: removeCommonWordsForWineMatch(nameReplaced)
            ,
        score: _helpers.getMatchSimilarity(
            removeCommonWordsForWineMatch(r.name),
             removeCommonWordsForWineMatch(nameReplaced)),
    });
});
```

```
match = _.orderBy(match, (e) => e.score.score, ['desc']);
match = _.first(match);
return match;
```

Após encontrar vinícola e vinho correspondentes, existe uma rotina simples que salva esses dados no banco, criando as relações mencionadas na Seção 3.1.3.

4.4 AVALIAÇÃO

Parte importante na hora de aperfeiçoar o algoritmo de pareamento de registros é verificar a eficiência do algoritmo. Para isso, a estratégia foi reunir uma amostra de cada vendedor e validar manualmente cada pareamento identificado. No primeiro momento, essa abordagem foi utilizada apenas para calibrar e testar o algoritmo, envolveu os próprios desenvolvedores do projeto e também outros avaliadores.

Nos casos de corrêspondencias incorretas, foi analisado o vinho correto, a fim de entender que parte do algoritmo poderia ser alterado para que ele acertasse. Foram feitas diversas rodadas de avaliação, conforme o algoritmo foi sendo melhorado.

Foi graças a essas rodadas de calibragem que foi encontrada a situação na qual o nome da vinícola do vendedor melhor correspondia com o nome do vinho do Vivino do que com o nome da vinícola do Vivino (como explicado em 4.3) e também a necessidade de atribuir pesos ao coeficiente de Sørensen–Dice e o algoritmo de Damerau-Levenshtein.

Após a calibragem, para de fato avaliar os resultados obtidos pela estratégia final proposta, era necessário remover o *viés* que poderia ser gerado caso a avaliação fosse feita pelos próprios desenvolvedores do projeto. Para tanto, foram coletadas amostras reduzidas, com vinhos de todos os vendedores, que foram distribuídas entre 6 avaliadores diferentes que deveriam indicar se os vinhos pareados pela estratégia proposta estavam corretos ou não.

Na avaliação final, cada avaliador recebeu uma planilha em formato eletrônico contendo 30 pares de vinhos (totalizando 180 pares), sobre os quais foram disponi-

bilizadas as seguintes informações:

- URL Vivino
- URL Revendedora
- Similaridade dos vinhos
- Similaridade das vinícolas
- *Match*
- Observação

As 4 primeiras colunas enviadas já estavam preenchidas, as 2 últimas foram preenchidas pelos avaliadores, onde eles colocavam 0 ou 1 na coluna *Match*, e caso não fosse uma correspondência (*Match* igual a 0), eles poderiam colocar uma observação contendo a URL do vinho correto no Vivino, caso fosse encontrado ou alguma observação que achasse pertinente (como, por exemplo, apontar que o algoritmo não acertou a vinícola e listar qual seria a vinícola correta). Um exemplo da planilha de avaliação está apresentada no Apêndice G.

A avaliação foi feita de forma manual, onde o usuário precisava abrir no seu navegador a URL do Vivino e da revendedora, analisar as informações exibidas e decidir se a correspondência estava correta ou não. Caso não estivesse, foi requisitado que o usuário tentasse encontrar no Vivino qual era a URL da correspondência correta e preenchesse na planilha.

Nesta avaliação da versão final da proposta (180 vinhos avaliados), realizada pelos avaliadores, houve uma taxa de acerto de aproximadamente 73%. Todavia, a taxa foi considerada satisfatória visto que os casos de não correspondência possuíam muitos problemas, como por exemplo os apontados na Seção 4.3.

Cabe destacar que toda a implementação da proposta foi executada utilizando a versão gratuita da *Amazon Web Services* (AWS)[5]. A máquina utilizada foi uma

[5]https://aws.amazon.com/free

"t2.micro", que possui 1 vCPU e 1 GiB de memória RAM. O banco de dados está hospedado também na AWS, utilizando o RDS (*Relational Database Service*) do tipo "db.t2.micro" que também possui 1 vCPU e 1 GiB de memória RAM.

5 DISPONIBILIZAÇÃO E EXIBIÇÃO DE DADOS

O propósito deste capítulo é apresentar e descrever quais foram as principais tecnologias utilizadas na implementação do projeto no que diz respeito à obtenção e disponibilização de dados a partir do banco de dados (descrito no Capítulo 3), já populado com os resultados do pareamento de registros (descrito no Capítulo 4), e exibição destes para o usuário. Para a obtenção e disponibilização dos dados para aplicações, foi construída uma API REST (*Representational State Transfer*). Para exibição dos dados, foi desenvolvida uma aplicação Web que permite a exibição da lista de vinhos, uso de diferentes filtros para seleção dos vinhos a serem exibidos e também a recomendação de vinhos ao usuário. A Figura 10 mostra o fluxo de obtenção de dados para exibição pela aplicação Web. Primeiramente, a API REST recebe requisições conforme demanda do usuário utilizando a aplicação Web. A API então faz a devida consulta ao banco de dados e retorna as informações para que a aplicação Web possa exibir os resultados para o usuário. As informações são retornadas do banco de dados para a API, convertidas para o formato JSON e carregadas para o usuário no *front end*.

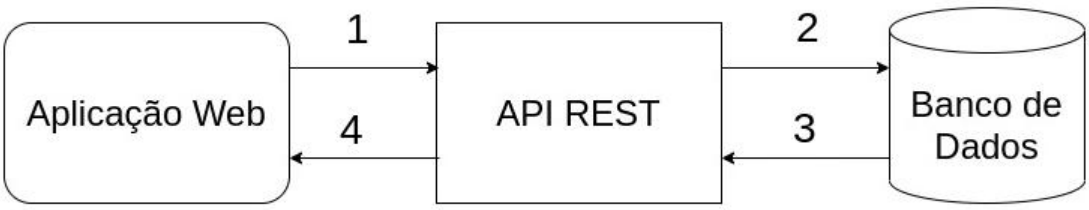

Figura 10: Fluxo de obtenção de dados pela aplicação Web via API

Para obter os dados e disponibilizá-los, a escolha foi a linguagem JavaScript utilizando a plataforma Node.js[1] em conjunto com o *framework* Express. Para o *front end*, foi utilizado o *framework* Angular[2], que utiliza TypeScript[3], um *superset* da linguagem JavaScript que, por criar uma camada de transpilação. Por conta disso, é possível identificar erros no código antes de sua execução, o que adiciona

[1] https://www.nodejs.org/
[2] https://angular.io/
[3] https://www.typescriptlang.org/

mais legibilidade e produtividade ao projeto.

5.1 API

Antes de pensar em exibição de dados para o usuário, foi necessário pensar em como disponibilizá-los para consumo, ou seja, como servir os dados de alguma maneira. Para tal, foi implementada uma API seguindo a arquitetura REST. Como esta parte do projeto foi desenvolvida utilizando Node.js, foi possível através de uma única *thread* trabalhar de maneira assíncrona, ou seja, é possível deixar essa única *thread* não bloqueada enquanto operações de entrada e saída são executadas.

Para simplificar o projeto da API, foi realizada a disponibilização dos dados do banco de dados através da geração de um ou mais arquivos JSON, e, por conta dessa padronização, a leitura e interpretação dos dados tornou-se mais viável. Para cada fonte de vinhos presente no banco de dados, foram criados um ou mais *endpoints* responsáveis por fazer a chamada do código necessário para efetuar consultas SQL (*Structured Query Language*) a fim de recuperar informações relevantes como nome, vinícola, imagens e outras informações dos vinhos.

Buscando separar a responsabilidade entre o código responsável por receber e responder as requisições HTTP e o acesso ao banco de dados, foi utilizado o padrão DAO (*Data access object*), o que significa que a lógica de acesso e consulta ao banco de dados estava separada do código responsável por receber as requisições, facilitando a leitura e manutenção posterior.

Para a comunicação entre a API e o banco de dados acontecer, foi criado um *pool* de conexões. Com isso, quando uma conexão é criada, ela é colocada no *pool* e pode ser usada novamente sem que uma nova conexão tenha que ser estabelecida. Caso haja múltiplos acessos de leitura simultâneos ao banco de dados, novas conexões serão criadas conforme a demanda simultânea for aumentando. Se um novo acesso ao banco de dados for iniciado e outro já tiver terminado, qualquer conexão que esteja liberada no *pool* poderá ser reutilizada.

Por conta da utilização de uma API REST neste trecho do projeto, houve total

desacoplamento e liberdade para escolha de qual tecnologia seria utilizada para exibir os dados para os usuários, pois a maneira de consumir os dados estava padronizada. Como descrito na Seção 5.2, foi utilizado o framework Angular, porém, caso houvesse interesse futuro em trocar a tecnologia usada para visualização dos dados, nada teria que ser mudado no código da API.

5.2 APLICAÇÃO WEB

Para a parte de visualização do usuário, foi utilizado o *framework* Angular, criado com o propósito de facilitar e abstrair conceitos para o desenvolvimento de aplicações *front end*. Um grande passo a ser tomado antes de iniciar o desenvolvimento da aplicação, foi decidir qual versão do Angular seria utilizada, já que, em suas primeiras versões, o *framework*, outrora chamado de AngularJS, utilizava a linguagem JavaScript como base, o que mudou a partir da versão 2, também chamada de Angular 2 ou Angular 2+. Como grande parte dos projetos modernos são desenvolvidos a partir da versão 2, foi decidido que seria utilizado o Angular em sua versão 9, que agora utiliza o *superset* da linguagem JavaScript, chamado TypeScript.

Além da experiência prévia dos alunos desenvolvendo o presente projeto, o Angular foi escolhido por conta de ser um dos maiores *frameworks* Web do mercado, e, também, por prover módulos que facilitam o envio de requisições e recebimento de respostas HTTP. Outro motivo de suma importância para a utilização do Angular, é a vasta gama de bibliotecas que provêm recursos de interface do usuário, autenticação e segurança.

A estrutura da aplicação Web foi dividida entre componentes e serviços. Os componentes são como pequenos blocos, que podem ser reutilizados, responsáveis por construir e montar os dados que serão exibidos para o usuário, ou seja, dentro do componente é feita toda a parte lógica de como as informações serão processadas e passadas para a visualização.

A aplicação foi dividida em dois principais componentes, um chamado *vivino-wines* e outro chamado *details*. A responsabilidade do componente *vivino-wines* é,

em posse da lista de vinhos obtidos, calcular e ajustar quantos serão exibidos por página, além de ser responsável por alterar os vinhos exibidos na tela ao mudar a página atual a ser exibida. O componente *details*, por sua vez, é responsável por construir a página de detalhes de um vinho, ao ser clicado pelo usuário.

Outro ponto para destacar é que, por conta da utilização do Angular, foi possível tornar a visualização ainda mais fácil de ser implementada. O *framework* provê uma série de *tags* próprias que podem ser utilizadas diretamente com HTML. Com isso, foi possível utilizar estruturas de repetição e condicional a fim de adequar aos diferentes cenários de exibição.

Dados modificados por alguma lógica no componente refletem suas mudanças imediatamente na visualização do usuário. Para que o Angular consiga identificar essas mudanças e altere a visualização, é utilizado um recurso chamado *two-way binding*. Um exemplo da utilização do *two-way binding* pode ser visto no Código 5.1, no qual, quando a lista de vinhos é preenchida ou alterada pelo componente, o mesmo pode ser manipulado diretamente na camada de visualização.

Código 5.1: Exemplo de repetição com two-way binding

```
<nb-card-body
   *ngFor="let item of pagesArray | paginate : { currentPage: page,
      itemsPerPage: 1 }; let i = index">
   <! -- Trecho omitido -->
</nb-card-body>
```

Por sua vez, os serviços são responsáveis por enviar requisições HTTP e obter os vinhos, ou seus detalhes, antes das páginas carregarem. Toda e qualquer informação utilizada por um componente é obtida através da utilização dos serviços, que através do módulo *http* do Angular, fazem o envio de requisições para a API, que, por sua vez, faz acesso ao banco de dados.

Nas seções seguintes, serão apresentados detalhes acerca das funcionalidades desenvolvidas para exibição dos dados sobre os vinhos aos usuários.

5.2.1 Tela Inicial

De início, é apresentada a tela inicial da aplicação, na qual o usuário pode navegar por mais de 50.000 páginas. Esta primeira abordagem tem como fim exibir um grande catálogo de visualização, sem qualquer critério de filtragem específico, possibilitando o conhecimento de novos vinhos para o usuário, como visto na Figura 11.

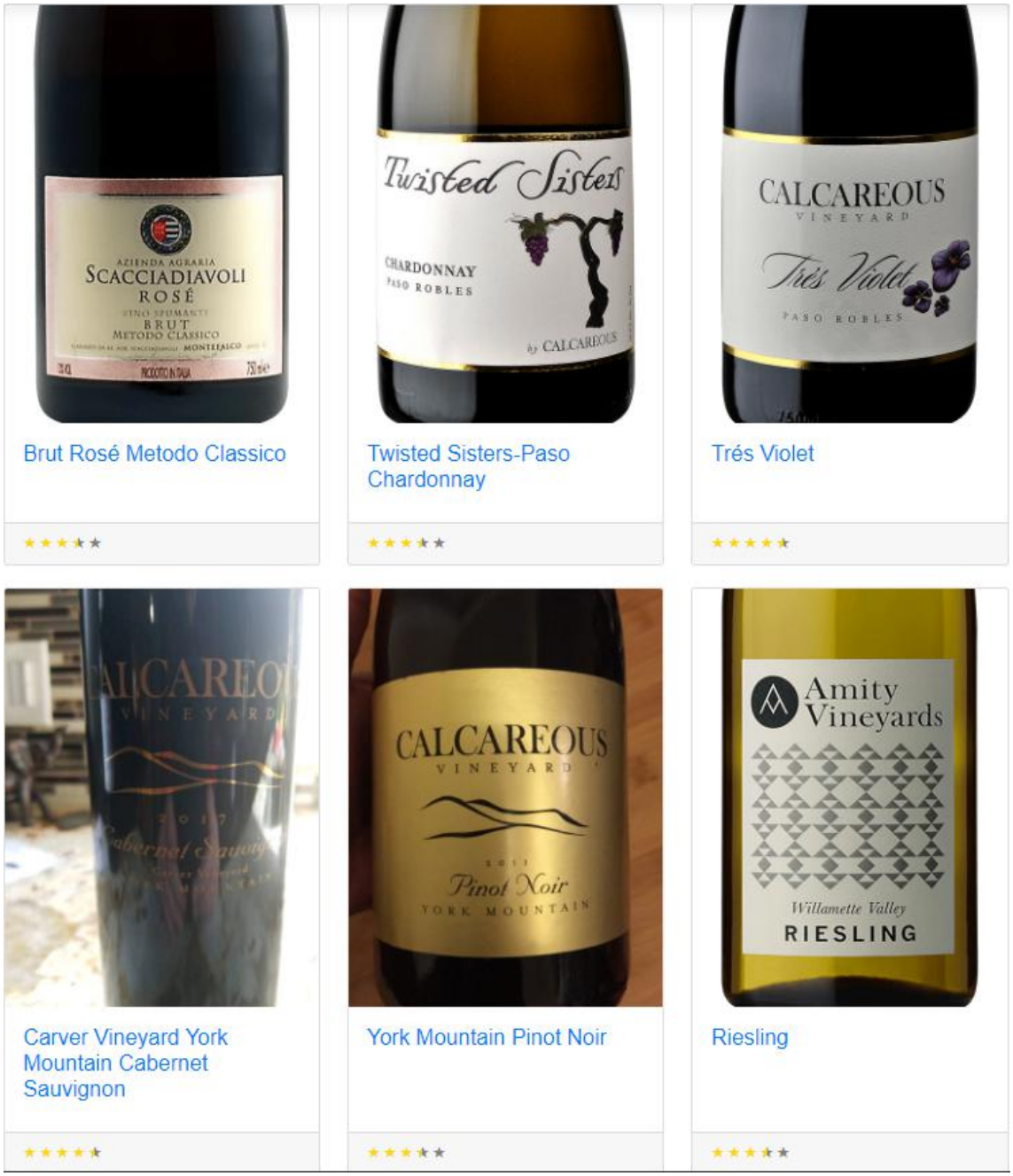

Figura 11: Exibição de vinhos na página inicial

Para trocar a página atual de vinhos, é necessário que o usuário navegue até o rodapé da mesma e selecione a próxima página, como pode ser visto na Figura 12.

Figura 12: Rodapé com seletor de páginas

5.2.2 Tela de Filtragem

Visando possibilitar uma seleção mais criteriosa por parte de usuários mais experientes e diminuir a sobrecarga de informação, auxiliando o usuário a encontrar vinhos de seu interesse, o sistema permite aplicar critérios de filtragem específicos. Para isso, a aplicação dispõe de uma tela onde o usuário pode definir filtros a fim de buscar vinhos segundo os critérios estabelecidos. Filtros por: (i) Uvas, (ii) Avaliação (nota) mínima, (iii) Países e (iv) Tipos de vinho, podem ser selecionados para que sejam exibidos vinhos de acordo com os interesses do usuário.

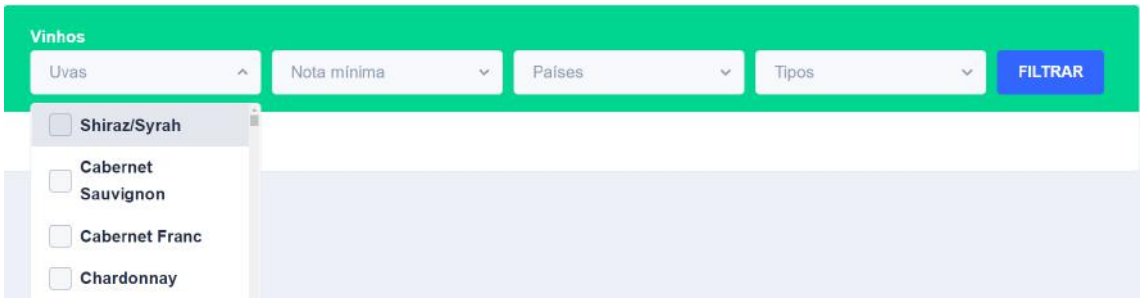

Figura 13: Tela de aplicação de filtros

5.2.3 Tela de Exibição de um Vinho Específico

Para garantir uma melhor experiência do usuário, cada vinho exibido possui um *link* responsável por redirecionar para uma página com mais detalhes sobre o vinho escolhido. Informações como tipo de uva, tipo do vinho e vinícola são exibidas para que o usuário possa tomar uma decisão mais assertiva.

Confirmando o interesse, o usuário poderá navegar diretamente para a página do Vivino ou da respectiva importadora que pode possuir o vinho em estoque, como pode ser visto na Figura 14.

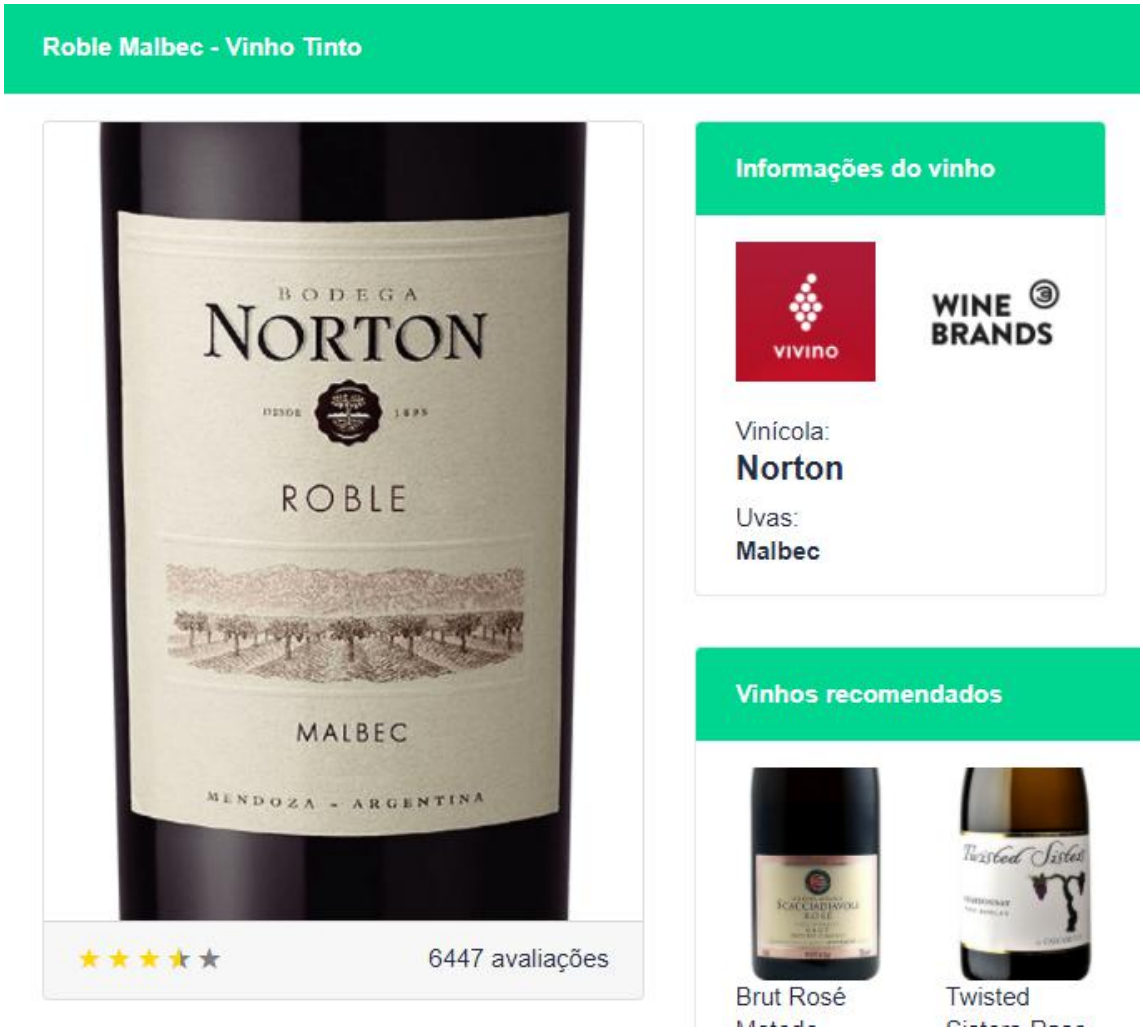

Figura 14: Página de detalhes de um vinho

Outra informação relevante exibida, é uma lista de vinhos recomendados baseada em características do vinho atual. Logo, caso o usuário escolha um vinho tinto

feito com uvas Malbec, a aplicação buscará vinhos com a mesma característica e os recomendará. Detalhes sobre a abordagem desenvolvida para recomendação de vinhos serão descritos na próxima seção (Seção 5.2.4).

5.2.4 Recomendação

O sistema de recomendação desenvolvido neste projeto utiliza Filtragem baseada em Conteúdo (2.3.2). Foi levantada uma gama de características que descrevem os vinhos que foram usadas para encontrar vinhos similares a serem recomendados ao usuário.

Dessa forma, a recomendação final implementada leva em consideração as seguintes características:

- Os dois vinhos com maior nota da vinícola atual
- Os dois vinhos com maior nota do mesmo tipo que o atual
- O vinho com maior nota do país atual

As escolhas realizadas levaram em consideração a simplicidade das estratégias implementadas, de maneira que pudessem ser rapidamente calculadas de forma *online* (no momento em que a requisição é realizada). Inicialmente, não foi possível utilizar uma abordagem colaborativa uma vez que não tínhamos avaliações dos vinhos pelos usuários da nossa aplicação Web.

6 CONCLUSÃO

No presente projeto, foi desenvolvido um sistema que abordou o tema de integração de dados, visando integrar dados sobre vinhos coletados a partir de diferentes fontes, construindo uma base focada no Vivino e interligada a diferentes revendedoras. O projeto conta hoje com três revendedoras locais e está desenvolvido de tal forma que a adição futura de novas fontes é feita de maneira totalmente independente de outras, e de fácil manutenção. Para que isto fosse possível, foram explorados, no desenvolvimento da solução, temas como coleta de dados na Web, pareamento de registros e sistemas de recomendação.

O desenvolvimento deste projeto proporcionou aos autores uma chance de aplicar diversos conteúdos estudados durante toda a graduação, passando por Computação 1 e 2 (por causa da lógica de programação); Banco De Dados (modelação e criação da base); Engenharia de Software (gerenciamento e organização do projeto) e até disciplinas eletivas, como Recuperação e Informação (Coleta de Dados na Web e Sistemas de Recomendação); Tópicos Especiais em Bancos de Dados (Trabalhando com grandes volumes de dados); dentre outras.

Diversos obstáculos foram enfrentados no desenvolvimento do projeto. Um dos maiores foi a frequência com que os métodos desenvolvidos para a coleta dos dados precisavam ser atualizados. Algumas das fontes utilizadas mudaram a forma de disponibilização e estruturação dos dados. Além disso, uma das fontes mudou completamente a comunicação entre o seu *front end* e *back end* e então, foi necessário começar do zero a etapa de entender essa comunicação e desenvolver um método de coleta.

O volume de dados trabalhados também trouxe desafios adicionais, sendo necessário o desenvolvimento de métodos eficientes computacionalmente para lidar com estes. Há necessidade de uma maior avaliação em termos de desempenho da estratégia de pareamento de registros proposta. Especificamente, a funcionalidade de recomendação de vinhos acabou sendo focada inicialmente em uma abordagem mais simples para ser calculada de forma *online*.

O trabalho foi implementado de forma que a inserção de mais revendedoras possa ser feita com facilidade, expandindo ainda mais a base. No entanto, a lógica de atualização dos vinhos já existentes não foi implementada. Diante disso, o principal recurso a ser desenvolvido é a criação de uma maneira de percorrer a base atual e coletar novamente os dados. Com isso, poderão ser atualizados fatores como por exemplo - preço de vinhos, avaliações, entre outros. Destarte, poderão buscar também novos vinhos que forem adicionados às fontes já existentes.

Ademais, podem ser estudadas futuras melhorias relativas à proposta de pareamento de registros de vinho como experimentos para avaliar um limiar (*threshold*) aplicado ao *score* para a identificação dos pares e aplicação de outras funções de similaridade e/ou uso de novos campos sobre os registros. Outro trabalho futuro interessante é explorar a aplicação de outras técnicas para aprimorar o sistema de recomendação. Para isso, um objetivo seria desenvolver uma biblioteca própria que suporte um cálculo mais eficiente na base de dados do Vivino e revendedoras. Outras possibilidades de expansão seriam não apenas recomendar vinhos, mas também vinícolas e regiões.

REFERÊNCIAS

[1] Abdul-Jabbar, S. S., e George, L. E. A comparative study for string metrics and the feasibility of joining them as combined text similarity measures. *ARO-THE SCIENTIFIC JOURNAL OF KOYA UNIVERSITY 5*, 2 (Nov. 2017), 6–18.

[2] Baeza-Yates, R., e Ribeiro-Neto, B. *Recuperação de Informação: Conceitos e Tecnologia das Máquinas de Busca*, 2nd ed. Bookman, 2013.

[3] Bilenko, M. Y. *Learnable Similarity Functions and Their Application to Record Linkage and Clustering record linkage.* PhD thesis, The University of Texas at Austin, 2006.

[4] Burke, R. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction 12*, 4 (2002), 331–370.

[5] Christen, P. *Data Matching - Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection.* Data-Centric Systems and Applications. Springer, 2012.

[6] Damerau, F. A technique for computer detection and correction of spelling errors. *Commun. ACM 7*, 3 (1964), 171–176.

[7] de Camargo Jr., K. R., e Coeli, C. M. Reclink: aplicativo para o relacionamento de bases de dados, implementando o método probabilistic record linkage. *Cad. Saúde Pública 16*, 2 (2000).

[8] Dice, L. R. Measures of the amount of ecologic association between species. *Ecology 26*, 3 (July 1945), 297–302.

[9] Dong, X. L., e Srivastava, D. *Big Data Integration.* Morgan & Claypool Publishersg, Berlin, Heidelberg, 2015.

[10] dos Santos Filho, W. Algoritmo paralelo e eficiente para o problema de pareamento de dados. Tese de Mestrado, Universidade Federal de Minas Gerais, Belo Horizonte. Orientador: Wagner Meira Junior.

[11] Fellegi, I. P., e Sunter, A. B. A theory of record linkage. *J Am Stat Assoc* (1969).

[12] Heuser, C. A. *Projeto de Banco de Dados*. Bookman, Porto Alegre, 2009.

[13] Jannach, D., Zanker, M., Felfernig, A., e Friedrich, G. *Recommender Systems: An Introduction*, 1st ed. Cambridge University Press, USA, 2010.

[14] Kallio, A. Automated web store product scraping using node.js. Tese de Mestrado, Tampere University of Technology, 2015.

[15] Kotonya, N., Cristofaro, P. D., e Cristofaro, E. D. Of wines and reviews: Measuring and modeling the vivino wine social network. *CoRR abs/1804.10982* (2018).

[16] Levenshtein, V. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady 10* (1966), 707.

[17] Lops, P., de Gemmis, M., e Semeraro, G. Content-based recommender systems: State of the art and trends. In *Recommender Systems Handbook*, F. Ricci, L. Rokach, B. Shapira, e P. B. Kantor, Eds. Springer, 2011, pp. 73–105.

[18] Mitchell, R. *Web Scraping with Python: Collecting Data from the Modern Web*. O'Reilly Media, 2015.

[19] Oliveira, L., Rocha Silva, R., e Bernardino, J. Wine ontology influence in a recommendation system. *Big Data and Cognitive Computing 5*, 2 (2021).

[20] Schafer, J. B., Frankowski, D., Herlocker, J., e Sen, S. *Collaborative Filtering Recommender Systems*. Springer-Verlag, Berlin, Heidelberg, 2007, p. 291–324.

[21] Schuring, R. Wine embeddings and a wine recommender, 2019. Disponível em: <https://towardsdatascience.com/robosomm-chapter-3-wine-embeddings-and-a-wine-recommender-9fc678f1041e>.

[22] Sørensen, T. *A method of establishing groups of equal amplitude in plant sociology based on similarity of species content and its application to analyses of the vegetation on Danish common*. København, I kommission hos E. Munksgaard, 1948.

[23] Zhu, Y., Matsuyama, Y., Ohashi, Y., e Setoguchi, S. When to conduct probabilistic linkage vs. deterministic linkage? a simulation study. *Journal of Biomedical Informatics 56* (2015), 80–86.

Apêndices

A DICIONÁRIO DO BANCO DE DADOS

Abaixo encontra-se dicionário das tabelas e colunas do banco de dados.

wineFromVivino	
Coluna	Descrição
id	identificador do vinho no Vivino
name	Nome do vinho, como retornado pela API do Vivino
url	URL do vinho no Vivino
fkType	Foreign Key para a tabela Type
fkRegio	Foreign Key para a tabela Region
fkStyle	Foreign Key para a tabela Style
fkWinery	Foreign Key para a tabela Winery
ratingsCount	Contagem de avaliações existentes para esse vinho no Vivino, informação retornada pela API
ratingsAvg	Média da nota das avaliações existentes para esse vinho no Vivino, informação retornada pela API
image	URL da imagem principal no Vivino, informação retornada pela API

As tabelas Style, Type e Grape possuem a mesma estrutura, suas informações são retornadas em cada vinho, as tabelas foram criadas apenas para normalizar o banco.

Style / Type / Grape	
Coluna	Descrição
id	identificador no Vivino
name	Nome, como retornado pela API do Vivino

Winery	
Coluna	Descrição
id	identificador da vinícola no Vivino
name	Nome, como retornado pela API do Vivino
ratingsCount	Contagem de avaliações existentes para essa vinícola no Vivino, informação retornada pela API
ratingsAvg	Média da nota das avaliações existentes para essa vinícola no Vivino, informação retornada pela API

Region	
Coluna	Descrição
id	identificador da região no Vivino
name	Nome, como retornado pela API do Vivino
countryCode	Código do país, como retornado pela API do Vivino
countryName	Nome do país, como retornado pela API do Vivino

A tabela wine_grape foi criada para relacionar os vinhos do Vivino com suas uvas, visto que era uma relação N:N.

wine_grape	
Coluna	Descrição
fkWine	Foreign Key para a tabela Wine
fkGrape	Foreign Key para a tabela Grape

wineFromSeller	
Coluna	Descrição
id	identificador auto-gerado
idFromSeller	identificador desse vinho no vendedor
name	Nome, como retornado pelo vendedor
fkSeller	Foreign Key para tabela wineSeller
url	URL do vinho no vendedor
image	URL da imagem no vendedor
fkType	Foreign Key para a tabela Type, baseado no algoritmo de comparação
fkRegio	Foreign Key para a tabela Region, baseado no algoritmo de comparação
fkWinery	Foreign Key para a tabela Winery, baseado no algoritmo de comparação
winery	Nome da vinícola, como retornado pelo vendedor
region	Região do vinho, como retornado pelo vendedor
json	JSON bruto de informações do vinho no vendedor

A tabela wineSeller é apenas para listar todos os vendedores que o projeto já possui rotina de extração de dados, e ligar com a tabela wineFromSeller

wineSeller	
Coluna	Descrição
id	identificador auto-gerado
name	Nome do vendedor

A tabela vivinoToSeller é preenchida baseada no algoritmo de comparação.

Winery	
Coluna	Descrição
fkWineFromVivino	Foreign Key para a tabela wineFromVivino
fkWineFromSeller	Foreign Key para a tabela wineFromSeller
winerySimilarity	Score de similaridade da vinícola do Vivino com a do vendedor, retornado pelo algoritmo de comparação
wineSimilarity	Score de similaridade do vinho do Vivino com o do vendedor, retornado pelo algoritmo de comparação
verifiedMatch	Coluna preenchida manualmente, quando o match foi verificado manualmente, 0 para resultado errado, 1 para correto, null para não verificado.

B JSON DE VINHOS DO VIVINO

Código B.1: JSON formatado com informações do Vivino


```
{
    "vintage": {
        "id": 1659071,
        "seo_name": "garibaldi-moscatel-nv",
        "name": "Garibaldi Moscatel",
        "statistics": {
            "status": "Normal",
            "ratings_count": 6493,
            "ratings_average": 4,
            "labels_count": 19320,
            "reviews_count": 1870
        },
        "organic_certification_id": null,
        "certified_biodynamic": null,
        "image": {
            "location": "//images.vivino.com/thumbs/9
               xiX_RIERDSJSDbRWoirjg_pl_480x640.png",
            "variations": {
                "bottle_large": "//images.vivino.com/thumbs/9
                   xiX_RIERDSJSDbRWoirjg_pb_x960.png",
                "bottle_medium": "//images.vivino.com/thumbs/9
                   xiX_RIERDSJSDbRWoirjg_pb_x600.png",
                "bottle_medium_square": "//images.vivino.com/thumbs/9
                   xiX_RIERDSJSDbRWoirjg_pb_600x600.png",
                "bottle_small": "//images.vivino.com/thumbs/9
                   xiX_RIERDSJSDbRWoirjg_pb_x300.png",
                "bottle_small_square": "//images.vivino.com/thumbs/9
                   xiX_RIERDSJSDbRWoirjg_pb_300x300.png",
                "label": "//images.vivino.com/thumbs/9
                   xiX_RIERDSJSDbRWoirjg_pl_480x640.png",
                "label_large": "//images.vivino.com/thumbs/9
                   xiX_RIERDSJSDbRWoirjg_pl_375x500.png",
                "label_medium": "//images.vivino.com/thumbs/9
                   xiX_RIERDSJSDbRWoirjg_pl_150x200.png",
```

```
            "label_medium_square": "//images.vivino.com/thumbs/9
                xiX_RIERDSJSDbRWoirjg_pl_150x150.png",
            "label_small_square": "//images.vivino.com/thumbs/9
                xiX_RIERDSJSDbRWoirjg_pl_80x80.png",
            "large": "//images.vivino.com/thumbs/9
                xiX_RIERDSJSDbRWoirjg_pl_375x500.png",
            "medium": "//images.vivino.com/thumbs/9
                xiX_RIERDSJSDbRWoirjg_pl_150x200.png",
            "medium_square": "//images.vivino.com/thumbs/9
                xiX_RIERDSJSDbRWoirjg_pl_150x150.png",
            "small_square": "//images.vivino.com/thumbs/9
                xiX_RIERDSJSDbRWoirjg_pl_80x80.png"
        }
    },
    "wine": {
        "id": 1273385,
        "name": "Moscatel",
        "seo_name": "moscatel",
        "type_id": 3,
        "vintage_type": 1,
        "is_natural": false,
        "region": {
            "id": 1465,
            "name": "Serra Gaúcha",
            "name_en": "",
            "seo_name": "serra-gaucha",
            "country": {
                "code": "br",
                "name": "Brasil",
                "native_name": "Brasil",
                "seo_name": "brazil",
                "currency": {
                    "code": "BRL",
                    "name": "Brazil Reais",
                    "prefix": "R\$",
                    "suffix": null
                },
                "regions_count": 24,
```

```
            "users_count": 3662753,
            "wines_count": 13667,
            "wineries_count": 1383,
            "most_used_grapes": [
                {
                    "id": 2,
                    "name": "Cabernet Sauvignon",
                    "seo_name": "cabernet-sauvignon",
                    "has_detailed_info": true,
                    "wines_count": 733594
                },
                {
                    "id": 10,
                    "name": "Merlot",
                    "seo_name": "merlot",
                    "has_detailed_info": true,
                    "wines_count": 516473
                },
                {
                    "id": 5,
                    "name": "Chardonnay",
                    "seo_name": "chardonnay",
                    "has_detailed_info": true,
                    "wines_count": 547882
                }
            ]
        },
        "parent_id": null,
        "background_image": {
            "location": "//images.vivino.com/regions/
               backgrounds/QKz33czSRi2IZNmeB626Dw.jpg",
            "variations": {
                "large": "//thumbs.vivino.com/
                   region_backgrounds/
                   QKz33czSRi2IZNmeB626Dw_1280x760.jpg",
                "medium": "//thumbs.vivino.com/
                   region_backgrounds/
                   QKz33czSRi2IZNmeB626Dw_600x356.jpg"
```

```
                    }
                },
                "class": {
                    "typecast_map": {
                        "background_image": {},
                        "class": {}
                    }
                },
                "statistics": {
                    "wineries_count": 0,
                    "wines_count": 0,
                    "sub_regions_count": 0,
                    "parent_regions_count": 0
                }
            },
            "review_status": 2,
            "winery": {
                "id": 29186,
                "name": "Garibaldi",
                "seo_name": "garibaldi",
                "status": 0,
                "review_status": "Completed",
                "statistics": {
                    "ratings_count": 25881,
                    "ratings_average": 3.7,
                    "labels_count": 111333,
                    "wines_count": 36
                },
                "region": {
                    "id": 1465,
                    "name": "Serra Gaúcha",
                    "name_en": "",
                    "seo_name": "serra-gaucha",
                    "country": {
                        "code": "br",
                        "name": "Brasil",
                        "native_name": "Brasil",
                        "seo_name": "brazil",
```

```
                "currency": {
                    "code": "BRL",
                    "name": "Brazil Reais",
                    "prefix": "R\$",
                    "suffix": null
                },
                "regions_count": 24,
                "users_count": 3662753,
                "wines_count": 13667,
                "wineries_count": 1383,
                "most_used_grapes": [
                    {
                        "id": 2,
                        "name": "Cabernet Sauvignon",
                        "seo_name": "cabernet-sauvignon",
                        "has_detailed_info": true,
                        "wines_count": 733594
                    },
                    {
                        "id": 10,
                        "name": "Merlot",
                        "seo_name": "merlot",
                        "has_detailed_info": true,
                        "wines_count": 516473
                    },
                    {
                        "id": 5,
                        "name": "Chardonnay",
                        "seo_name": "chardonnay",
                        "has_detailed_info": true,
                        "wines_count": 547882
                    }
                ]
            },
            "parent_id": null,
            "background_image": {
                "location": "//images.vivino.com/regions/
                   backgrounds/QKz33czSRi2IZNmeB626Dw.jpg",
```

```
                    "variations": {
                        "large": "//thumbs.vivino.com/
                            region_backgrounds/
                            QKz33czSRi2IZNmeB626Dw_1280x760.jpg",
                        "medium": "//thumbs.vivino.com/
                            region_backgrounds/
                            QKz33czSRi2IZNmeB626Dw_600x356.jpg"
                    }
                },
                "class": {
                    "typecast_map": {
                        "background_image": {},
                        "class": {}
                    }
                },
                "statistics": {
                    "wineries_count": 0,
                    "wines_count": 0,
                    "sub_regions_count": 0,
                    "parent_regions_count": 0
                }
            },
            "business_name": "Cooperativa Vinicola Garibaldi",
            "description": "",
            "specialists_notes": "",
            "phone": "54 3464 8100",
            "email": "",
            "facebook": "",
            "instagram": "",
            "is_claimed": false,
            "twitter": "",
            "website": "http://www.vinicolagaribaldi.com.br",
            "winemaker": null,
            "wine_maker": null,
            "address": {
                "title": null,
                "name": null,
                "street": "Av. Independência, 845 - Centro",
```

```
                "street2": null,
                "neighborhood": null,
                "city": "Garibaldi",
                "zip": " 95720-000",
                "state": null,
                "country": {
                    "code": "br",
                    "name": "Brasil",
                    "native_name": "Brasil",
                    "seo_name": "brazil",
                    "currency": {
                        "code": "BRL",
                        "name": "Brazil Reais",
                        "prefix": "R\$",
                        "suffix": null
                    },
                    "regions_count": 24,
                    "users_count": 3662753,
                    "wines_count": 13667,
                    "wineries_count": 1383,
                    "most_used_grapes": [
                        {
                            "id": 2,
                            "name": "Cabernet Sauvignon",
                            "seo_name": "cabernet-sauvignon",
                            "has_detailed_info": true,
                            "wines_count": 733594
                        },
                        {
                            "id": 10,
                            "name": "Merlot",
                            "seo_name": "merlot",
                            "has_detailed_info": true,
                            "wines_count": 516473
                        },
                        {
                            "id": 5,
                            "name": "Chardonnay",
```

```
                        "seo_name": "chardonnay",
                        "has_detailed_info": true,
                        "wines_count": 547882
                    }
                ]
            },
            "company": null,
            "phone": null,
            "external_id": null,
            "residential": null,
            "vat_number": null,
            "vat_code": null,
            "addition": null
        },
        "image": null,
        "location": {
            "latitude": -29.257458,
            "longitude": -51.52322
        },
        "winery_group": null,
        "first_wines": []
    },
    "statistics": {
        "status": "Normal",
        "ratings_count": 6493,
        "ratings_average": 4,
        "labels_count": 25469,
        "vintages_count": 32
    },
    "rank": {
        "country": {
            "description": "Classificaç o  no(a) Brasil",
            "total": 9033,
            "rank": 156
        },
        "region": {
            "description": "Classificaç o  na regi o do vinho"
                ,
```

```
                    "total": 4086,
                    "rank": 77
                },
                "winery": {
                    "description": "Classificaç o  na vinícola",
                    "total": 34,
                    "rank": 2
                },
                "wine_type": {
                    "description": "Rank within all Vinho espumante",
                    "total": 98735,
                    "rank": 2063
                },
                "global": {
                    "description": "Classificaç o  global",
                    "total": 1213955,
                    "rank": 32332
                }
            }
        }
    }
}
```

C JSON DE VINHOS DO WINEBRANDS

Código C.1: JSON formatado com informações do Winebrands


```
{
    "Uva": [
        "Chardonnay"
    ],
    "Tipo": [
        "Doce e/ou Fortificado"
    ],
    "link": "https://www.winebrands.com.br/vinho-de-sobremesa-norton-
       cosecha-tardia-branco-2018/p",
    "País": [
        "Argentina"
    ],
    "Safra": [
        "2018"
    ],
    "brand": "Norton",
    "items": [
        {
            "ean": "",
            "name": "750ml",
            "isKit": false,
            "Videos": [],
            "Volume": [
                "750ml"
            ],
            "images": [
                {
                    /...
                }
            ],
            "itemId": "2002838",
            "sellers": [
                {
                    "sellerId": "1",
```

```
                "sellerName": "Winebrands",
                "addToCartLink": "https://www.winebrands.com.br/
                    checkout/cart/add?sku=2002838&qty=1&seller=1&sc
                    =1&price=6500&cv=
                    b9ae21c9e3c2f772c1da908f858bdd0e_&sc=1",
                "sellerDefault": true,
                "commertialOffer": {
                    "Tax": 0,
                    "Price": 65,
                    "ListPrice": 65,
                    "GiftSkuIds": [],
                    "BuyTogether": [],
                    "RewardValue": 0,
                    "Installments": [
                        /...
                    ],
                    "PaymentOptions": {
                        /...
                        ],
                        /...
                        ]
                    },
                    /...
                }
            }
        ],
        "modalType": null,
        "variations": [
            "Volume"
        ],
        "referenceId": [
            {
                "Key": "RefId",
                "Value": "NT0231800"
            }
        ],
        "nameComplete": "Vinho De Sobremesa - Cosecha Tardia Branco
            2018 750ml",
```

```
            "complementName": "<p class=\"pais\"><em class=\"bd ar\"></
                em><strong>Argentina - Mendoza, Lujan de Cuyo | Norton
                </strong> </p> <p class=\"dc\"> <strong>Dica do
                Sommelier </strong> Elaborado com uvas Chardonnay
                colhidas tardiamente,  e ricas em açúcar, este vinho
                doce natural possui encantadoras notas de mel e frutas
                brancas maduras. Um vinho que, dependendo das escolhas,
                 pode acompanhar uma refeiç o  inteira. Experimente com
                 queijos azuis, ou com uma salada de frutas fresquinhas
                .</p>",
            "unitMultiplier": 1,
            "measurementUnit": "un",
            "estimatedDateArrival": null
        }
    ],
    "Volume": [
        "750 ml"
    ],
    "Regi o": [
        "Mendoza"
    ],
    "brandId": 2000024,
    "Produtor": [
        "Norton"
    ],
    "linkText": "vinho-de-sobremesa-norton-cosecha-tardia-branco-2018",
    "productId": "2002728",
    "categories": [
        "/Vinhos/Argentina/Doce e Fortificado/",
        "/Vinhos/Argentina/",
        "/Vinhos/"
    ],
    "categoryId": "1000095",
    "Descriç o ": [
        /...
    ],
    "Sub-Regi o": [
        "Mendoza, Lujan de Cuyo"
```

```
    ],
    "description": "Vinho De Sobremesa - Safra 2018",
    "productName": "Vinho De Sobremesa - Cosecha Tardia Branco 2018",
    "releaseDate": "2015-10-07T00:00:00",
    "productTitle": "Cosecha Tardia | Vinhos argentinos",
    "brandImageUrl": null,
    "categoriesIds": [
        "/1000009/1000070/1000095/",
        "/1000009/1000070/",
        "/1000009/"
    ],
    "Especificaç o ": [
        "Descriç o ",
        "Tipo",
        "País",
        "Regi o",
        "Produtor",
        "Uva",
        "Volume",
        "Safra",
        "Sub-Regi o"
    ],
    "productClusters": {
        "140": "teste profite nosso nosso",
        "224": "OPORTUNIDADES",
        "294": "Todos os Produtos",
        "411": "Marketplace",
        "517": "Vinhos Doces e Fortificados",
        "520": "voce tambem pode gostar"
    },
    "productReference": "NT0231800",
    "allSpecifications": [
        "Descriç o ",
        "Tipo",
        "País",
        "Regi o",
        "Produtor",
        "Uva",
```

```
        "Volume",
        "Safra",
        "Sub-Regi o"
    ],
    "clusterHighlights": {},
    "skuSpecifications": [
        {
            "field": {
                "id": 30,
                "name": "Volume",
                "type": "Combo",
                "isActive": true,
                "position": 2
            },
            "values": [
                {
                    "id": "539",
                    "name": "750ml",
                    "position": 3
                }
            ]
        }
    ],
    "metaTagDescription": "Vinho De Sobremesa - Cosecha Tardia Branco
       2018 | Winebrands é Importadora, Distribuidora e Loja Online de
        Vinhos Referência no Segmento!",
    "searchableClusters": {
        "224": "OPORTUNIDADES",
        "517": "Vinhos Doces e Fortificados",
        "520": "voce tambem pode gostar"
    },
    "allSpecificationsGroups": [
        "Especificaç o "
    ]
}
```

D JSON DE VINHOS DO GRAND CRU

Código D.1: JSON formatado com informações do Grand Cru


```
{
    "id": 5048689229883,
    "tags": [
        "América do Sul",
        "Chile",
        "Churrasco",
        "Dia-a-dia",
        "Encorpado",
        "Entre R$ 100 e R$ 200",
        "Equilibrado",
        "Frutado",
        "GrandCru-Momento-de-Consumo-Dia-a-dia Churrasco",
        "GrandCru-Pais-Chile",
        "GrandCru-precocheio",
        "GrandCru-Tipo-de-vinho-Tinto",
        "GrandCru-Tipo-VINHOS",
        "grandcru-uva-Carmenere",
        "GrandCru-Vinho-como-eu-vejo-Frutado Equilibrado Encorpado",
        "GrandCru-Vinhos-Organicos-N o",
        "GrandCru-Vinicola-Viña Mancura",
        "Novo Mundo",
        "Tinto",
        "VINHOS",
        "Viña Mancura"
    ],
    "title": "Mancura Mito Carmenere 2017 750 mL",
    "handle": "vinho-mancura-mito-carmenere-2017-750-ml",
    "images": [
        {
            "id": 16972368314427,
            "src": "https://cdn.shopify.com/s/files/1/0301/7405/2411/
               products/CHMAN0909A17-vinho-tinto-mancura-mito-
               carmenere-2017-750ml.png?v=1602785419",
            "width": 259,
```

```
            "height": 691,
            "position": 1,
            "created_at": "2020-10-15T14:53:37-03:00",
            "product_id": 5048689229883,
            "updated_at": "2020-10-15T15:10:19-03:00",
            "variant_ids": []
        }
    ],
    "vendor": "Vina Mancura",
    "options": [
        {
            "name": "Title",
            "values": [
                "Default Title"
            ],
            "position": 1
        }
    ],
    "variants": [
        {
            "id": 34251162255419,
            "sku": "CHMAN0909A17",
            "grams": 0,
            "price": "149.90",
            "title": "Default Title",
            "option1": "Default Title",
            "option2": null,
            "option3": null,
            "taxable": true,
            "position": 1,
            "available": true,
            "created_at": "2020-10-15T13:38:53-03:00",
            "product_id": 5048689229883,
            "updated_at": "2020-10-19T12:38:27-03:00",
            "featured_image": null,
            "compare_at_price": null,
            "requires_shipping": true
        }
```

```
    ],
    "body_html": "",
    "created_at": "2020-10-15T13:38:53-03:00",
    "updated_at": "2020-10-19T12:38:27-03:00",
    "product_type": "VINHO",
    "published_at": "2020-10-19T09:23:03-03:00",
    "crawledInformation": {
        "uva": "100% Carmenere",
        "safra": "2017",
        "regiao": "Vale do Maule",
        "visual": "Vermelho com tonalidade granada",
        "tamanho": "750 ml",
        "olfativo": "Aromas de frutas negras com toque de especiarias",
        "vinicola": "Viña Mancura",
        "como-vejo": "Frutado, Equilibrado, Encorpado",
        "gustativo": "Em boca um vinho de muita concentraç o  e volume
           com taninos sedosos",
        "tipo-vinho": "Tinto",
        "momento-consumo": "Dia-a-dia, Churrasco"
    }
}
```

E JSON DE VINHOS DO EVINO

Código E.1: JSON formatado com informações do Evino


```
{
    "data": {
      "meta": {
        "perPage": 10,
        "pages": {
          "active": 1,
          "first": 1,
          "prev": null,
          "next": 2,
          "last": 20
        },
        "total": 194
      },
      "products": [
        {
          "sku": "1671590",
          "name": "La Chanson du Soleil Cuvée Grande Réserve",
          "validForSale": true,
          "quantity": 2011,
          "isBundle": false,
          "images": {
            "thumbnail": "https://res.cloudinary.com/evino/image/upload
               /c_scale,f_auto,fl_progressive,q_auto:eco,fl_lossy,
               q_auto,c_pad,w_18/products/1671590-standing-front.jpg",
            "small": "https://res.cloudinary.com/evino/image/upload/
               c_scale,f_auto,fl_progressive,q_auto:eco,fl_lossy,
               q_auto,c_pad,w_36/products/1671590-standing-front.jpg",
            "medium": "https://res.cloudinary.com/evino/image/upload/
               c_scale,f_auto,fl_progressive,q_auto:eco,fl_lossy,
               q_auto,c_pad,h_350/products/1671590-standing-front.jpg"
               ,
            "large": "https://res.cloudinary.com/evino/image/upload/
               c_scale,f_auto,fl_progressive,q_auto:eco,fl_lossy,
```

```
            q_auto,c_pad,w_160/products/1671590-standing-front.jpg"
            ,
          "extralarge": "https://res.cloudinary.com/evino/image/
            upload/c_scale,f_auto,fl_progressive,q_auto:eco,
            fl_lossy,q_auto,c_pad,w_675/products/1671590-standing-
            front.jpg",
          "fallback": {
            "thumbnail": "https://d2vf6jska6oose.cloudfront.net/BR/
              product/la-chanson-du-soleil-rouge
              -74681-3216-18647-3-675w.jpg",
            "small": "https://d2vf6jska6oose.cloudfront.net/BR/
              product/la-chanson-du-soleil-rouge
              -74681-3216-18647-3-675w.jpg",
            "medium": "https://d2vf6jska6oose.cloudfront.net/BR/
              product/la-chanson-du-soleil-rouge
              -74681-3216-18647-3-675w.jpg",
            "large": "https://d2vf6jska6oose.cloudfront.net/BR/
              product/la-chanson-du-soleil-rouge
              -74681-3216-18647-3-675w.jpg",
            "extralarge": "https://d2vf6jska6oose.cloudfront.net/BR/
              product/la-chanson-du-soleil-rouge
              -74681-3216-18647-3-675w.jpg"
          }
        },
        "url": "la-chanson-du-soleil-rouge-74681.html",
        "type": "Tinto",
        "expiration": "2017-07-22T02:59:59-0300",
        "whyThisWine": "",
        "aboutThisWine": "Les Cinq é um exemplar versátil e muito
          saboroso, perfeito para vários momentos da vida. Mas em
          especial, a pizza de domingo - ela pede por um tinto,
          n o é mesmo? Mas se você vai trabalhar na segunda cedo
          ou n o gosta muito de pizza, este rótulo certamente se
          encaixa na sua rotina, afinal: há sempre espaço para um
          bom vinho.",
        "servingTemperature": "16 C ",
        "closureType": "Rolha de cortiça",
        "alcoholContent": "12,5%",
```

```
        "pairing": "Pizzas, massas e queijos",
        "intensity": "",
        "volume": "750mL",
        "prices": {
          "recommended": 6480,
          "sale": 2990,
          "subscriber": null
        },
        "discounts": {
          "sale": 54,
          "subscriber": null
        },
        "producer": {
          "id": null,
          "name": "Vignobles Roux",
          "country": "França",
          "region": "Várias regi es",
          "quantity": null
        },
        "mainCampaign": 18301,
        "campaigns": [
          {
            "id": 31,
            "url": "http://api.evino.com.br/campaign/31"
          },
          {
            "id": 12822,
            "url": "http://api.evino.com.br/campaign/12822"
          },
          {
            "id": 18301,
            "url": "http://api.evino.com.br/campaign/18301"
          }
        ],
        "parentBundles": {
          "sku": "031511"
        },
        "bundleProducts": [
```

```
        ],
        "sommelier": {
          "reviewer": "Jéssica Marinzeck",
          "commentary": "Um tinto com o equilíbrio perfeito entre
              intensidade e maciez, possui coloraç o  vermelho-rubi e
               aromas de frutas vermelhas como morango e framboesa. "
              ,
          "color": "Vermelho-rubi",
          "mouth": "Equilibrio perfeito entre maciez e intensidade.",
          "maturing": null,
          "nose": "Frutas vermelhas frescas",
          "keepUntil": 2019,
          "picture": null,
          "gender": "Female"
        },
        "grapes": [

        ],
        "countries": [
          {
            "name": "França",
            "icon": "https://static.evino.com.br/BR/upload/APP/
                content/country/icons/franca/franca@2x.png"
          }
        ],
        "harmonization": {
          "id": 31,
          "name": "Pizza e massas de molho vermelho",
          "description": null,
          "url": "http://api.evino.com.br/food/recipe/31"
        },
        "prizesMedals": null,
        "isExpired": false,
        "attributeSet": "Stillwine"
      }
}
```

F CÓDIGO DA FUNÇÃO GETWINEOBJECT

Código F.1: Parser de vinhos

```
async function getWineObject(seller, wine) {
    try {
        if (!seller.isWine(wine)) {
            return null;
        }
        const obj = {};
        obj.json = wine;
        obj.type = seller.getWineType(wine);
        obj.seller = seller.getSellerId();
        obj.image = seller.getWineImage(wine);
        obj.country = await translate(seller.getWineCountry(wine), {
            from: 'pt',
            to: 'en',
        })
            .then((response) => {
                console.log(response.data);
                if (response.data === 'U.S' || response.data === 'USA')
                    {
                    response.data = 'United States';
                }
                return lookup.byCountry(response.data).country;
            })
            .catch((e) => {
                console.log(e.message);
            });
        obj.name = seller.getWineName(wine);
        obj.url = seller.getWineUrl(wine);
        try {
            const tempRegion = await translate(seller.getWineRegion(
               wine), {
                to: 'en',
            })
                .then((response) => response.data[0])
                .catch((e) => {
```

```
                    console.log(e.message);
                });
            obj.regionTranslated = tempRegion;
        } catch (e) {
            obj.regionTranslated = null;
        }
        obj.region = seller.getWineRegion(wine);
        obj.regionDb = await dbUtils.getRegionDb(
            seller.getWineRegion(wine),
            obj.regionTranslated,
            obj.country
        );
        obj.typeDb = await dbUtils.getTypeDb(obj.type);
        obj.winery = seller.getWineWinery(wine);
        obj.wineryDb = await dbUtils.getWineryDb(
            seller.getWineWinery(wine),
            obj.name,
            obj.country
        );
        obj.wineVivino = await dbUtils.getWineVivinoDb(
            obj.name,
            obj.typeDb,
            obj.wineryDb
        );
        return obj;
    } catch (e) {
        console.log(JSON.stringify(e.message));
        return null;
    }
}
```

G TABELA DE VALIDAÇÃO MANUAL

Abaixo encontra-se um exemplo de tabela enviada para os usuários que auxiliaram na etapa de validação manual.

URL Vendedor	URL Vivino	Simil. Vinícola	Simil. Vinho	Match	OBS
winebrands.com.br/ vinho-tinto-norton-privado-2014/p	vivino.com/ norton-privada/w/3208	1	0,83	1	
winebrands.com.br/ vinho-de-sobremesa-norton-cosecha-tardia-rose-2017/p	vivino.com/ norton-cosecha-tardia-blanco-dulce-lujan-de-cuyo/w/5866423	1	0,67	0	Vinho correto: vivino.com/ norton-cosecha-tardia-espumante-dulce-rose/w/2570043
winebrands.com.br/ muriel-vina-eguia-tempranillo-2015/p	vivino.com/ muriel-tempranillo-rioja/w/1378628	0,89	0,53	0	Vinícola no vivino tem outro nome. (vivino.com/eguia-tempranillo-rioja/ w/4801758)
evino.com.br/product/ marques-de-montino-tempranillo-132431.html	vivino.com/marques-del-atrio-atrio-swalero-vino-tinto-red-wine-v-pvite/w/4451398	1	0,35	0	Vinicola errada.(vivino.com/montino-tinto-castilla-la-mancha/w/6988564)
grandcru.com.br/products/vinho-branco-san-pedro-de-yacochuya-torrontes-2019-750-ml	vivino.com/san-pedro-de-yacochuya-torrontes/w/1105351	1	1	1	

H CÓDIGO DE-PARA

Em todos os códigos abaixo foi utilizada a variável "_"que é uma variável do tipo Lodash [1]

Código H.1: Evino

```
function getWineName(json) {
    let name = _.get(json, 'productName', _.get(json, 'productDetail.
        name', ''));
    name = name.replace(this.getWineYear(json), '');
    return name;
}
function getWineYear(json) {
    return _.get(json, 'productDetail.vintage', '');
}
function getWineUrl(json) {
    return 'https://www.evino.com.br/product/' +_.get(json, '
        productDetail.url', '');
}
function getWineBottleSize(json) {
    return _.get(json, 'productDetail.volume', '');
}
function isWine(json) {
    return _.get(json, 'productDetail.classification', '') === 'Wine';
}
function getWineType(json) {
    const type = _.get(json, 'productDetail.type', '');
    switch (_.toLower(type)) {
    case 'tinto':
        return 'Red Wine';
    case 'branco':
        return 'White Wine';
    case 'espumante branco':
        return 'Sparkling';
    case 'espumante rosé':
        return 'Sparkling';
```

[1] https://lodash.com/

```
    case 'rosé':
        return 'Rosé Wine';
    case 'rose':
        return 'Rosé Wine';
    case 'fortificado':
        return 'Fortified Wine';

    default:
        return type;
    }
}
function getWineCountry(json) {
    return _.get(json, 'productDetail.producer.country', _.get(json, '
       productDetail.countries[0].name'));
}
function getWineRegion(json) {
    return _.get(json, 'productDetail.producer.region');
}
function getWineSubRegion(json) {
    return null;
}
function getWineWinery(json) {
    return _.get(json, 'productDetail.producer.name');
}
function getWineId(json) {
    return _.get(json, 'productDetail.sku');
}
function getWineImage(json) {
    return _.get(json, 'productDetail.images.extralarge', _.get(json, '
       productDetail.images.large', _.get(json, 'productDetail.images.
       medium', '')));
}
function getSellerId() {
    return 2;
}
```

Código H.2: WineBrands

```
function getWineName(json) {
    const name = json.productName;
    return _.trim(name.split(' - ')[1].replace(this.getWineWinery(json)
        , ''));
}
function getWineYear(json) {
    return _.chain(json).get('Safra', []).first()
        .value() || null;
}
function getWineUrl(json) {
    return _.get(json, 'link');
}
function getWineBottleSize(json) {
    return _.chain(json).get('Volume', []).first()
        .value() || null;
}
function isWine(json) {
    return _.get(json, 'productName', '').startsWith('Vinho');
}
function getWineType(json) {
    const categories = _.get(json, 'categories');
    const type = _.chain(categories)
        .map((x) => x.split('/'))
        .orderBy([
            (x) => {
                _.size(x);
            },
            'desc',
        ])
        .first()
        .at(3)
        .value();
    switch (_.toLower(type)) {
    case 'tinto':
        return 'Red Wine';
    case 'branco':
```

```
        return 'White Wine';
    case 'espumante':
        return 'Sparkling';
    case 'rosé':
        return 'Rosé Wine';
    case 'doce e fortificado':
        return 'Dessert Wine';

    default:
        return type;
    }
}
function getWineCountry(json) {
    return _.chain(json).get('País', []).first()
        .value() || null;
}
function getWineRegion(json) {
    return _.chain(json).get('Regi o', []).first()
        .value() || null;
}
function getWineSubRegion(json) {
    return null;
}
function getWineWinery(json) {
    return _.chain(json).get('Produtor', []).first()
        .value() || null;
}
function getWineId(json) {
    return _.get(json, 'productId');
}
function getWineImage(json) {
    const name = json.productName;
    return _.get(json, 'items[0].images[0].imageUrl', '');
}
function getSellerId() {
    return 1;
}
```

Código H.3: Grand Cru

```
function getWineName(json) {
    let name = _.get(json, 'title', _.get(json, 'variants[0].name', '')
        );
    name = name.replace(`${this.getWineYear(json)}`, '').trim();
    name = name.replace(`${this.getWineBottleSize(json)}`, '').trim();
    return name;
}
function getWineYear(json) {
    return _.get(json, 'crawledInformation.safra', '');
}
function getWineUrl(json) {
    return `https://www.grandcru.com.br/products/${_.get(json, 'handle
        ', '')}`;
}
function getWineBottleSize(json) {
    // Os replaces s o pra quando vier a info junta (750ml),
        transformar em (750 ml)
    // E se ja vier separada, eu acabo separando mais e depois
        removendo espaços duplos
    const reg = new RegExp('ml', 'gi');
    let bottleSize = _.get(json, 'crawledInformation.tamanho', '');
    bottleSize = _helpers.replaceCaseInsensitive(bottleSize, 'ml', ' ml
        ');
    return bottleSize.replace('  ', ' ');
}
function isWine(wine) {
    return true;
}
function getWineType(json) {
    const tags = _.get(json, 'tags', '');
    const type = _.find(tags, (x) => x.includes('GrandCru-Tipo-de-vinho
        -')
    ).replace('GrandCru-Tipo-de-vinho-', '');
    switch (_.toLower(type)) {
    case 'tinto':
        return 'Red Wine';
```

```
    case 'branco':
        return 'White Wine';
    case 'espumante':
        return 'Sparkling';
    case 'rosé':
        return 'Rosé Wine';
    case 'sobremesa':
        return 'Dessert Wine';

    default:
        return type;
    }
}
function getWineCountry(json) {
    const tags = _.get(json, 'tags', '');
    return (
        _.find(tags, (x) => x.includes('GrandCru-Pais-')).replace(
            'GrandCru-Pais-',
            ''
        ) || null
    );
}
function getWineRegion(json) {
    return _.get(json, 'crawledInformation.regiao', '');
}
function getWineSubRegion(json) {
    return null;
}
function getWineWinery(json) {
    const tags = _.get(json, 'tags', '');
    return (
        _.find(tags, (x) => x.includes('GrandCru-Vinicola-')).replace(
            'GrandCru-Vinicola-',
            ''
        ) ||
        _.get(json, 'crawledInformation.vinicola', null) ||
        null
    );
```

```
}
function getWineId(json) {
    return _.get(json, 'id');
}
function getWineImage(json) {
    const name = json.productName;
    return _.get(json, 'images[0].src', '');
}
function getSellerId() {
    return 3;
}
```